Defense Threat Reduction Agency
8725 John J. Kingman Road, MS-6201
Fort Belvoir, VA 22060-6201

DTRA-TR-15-28

TECHNICAL REPORT

Combating Weapons of Mass Destruction: Models, Complexity, and Algorithms in Complex Dynamic and Evolving Networks

November 2015

HDTRA1-09-1-0061

Dr. My Thai

Prepared by:
University of Florida
1 University of Florida
Gainesville, FL 32611

DESTRUCTION NOTICE:

Destroy this report when it is no longer needed.
Do not return to sender.

PLEASE NOTIFY THE DEFENSE THREAT REDUCTION AGENCY, ATTN: DTRIAC/ J9STT, 8725 JOHN J. KINGMAN ROAD, MS-6201, FT BELVOIR, VA 22060-6201, IF YOUR ADDRESS IS INCORRECT, IF YOU WISH THAT IT BE DELETED FROM THE DISTRIBUTION LIST, OR IF THE ADDRESSEE IS NO LONGER EMPLOYED BY YOUR ORGANIZATION.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
00-11-2015	Technical	July 16, 2009 - August 31, 2014

4. TITLE AND SUBTITLE

Combating Weapons of Mass Destruction: Models, Complexity, and Algorithms in Complex Dynamic and Evolving Networks

5a. CONTRACT NUMBER
HDTRA1-09-1-0061

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

My Thai

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

University of Florida
1 University of Florida
Gainesville,FL 32611

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Threat Reduction Agency
8725 John J. Kingman Road
Fort Belvoir, VA 22060

10. SPONSOR/MONITOR'S ACRONYM(S)
DTRA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
DTRA-TR-15-28

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution Statement A. Approved for public release, distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This project considers attack and defense problems on networks with respect to WMD attacks. It provides novel optimization models and solutions for network vulnerability assessment and defense measurement in the face of cascading failures and dynamic attacks. The critical infrastructures considered are complex systems which consist of multiple dynamic independent networks interacting to each other. The attacks we considered are dynamic, that is, another attack may be launched during the recovery.

15. SUBJECT TERMS

WMD network node vulnerability terrorist network

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	SAR	604	Allison Lyalikov
Unclassified	Unclassified	Unclassified			**19b. TELEPHONE NUMBER** *(include area code)* 703-767-4663

CONVERSION TABLE

Conversion Factors for U.S. Customary to metric (SI) units of measurement.

MULTIPLY →	BY →	TO GET
TO GET ←	BY ←	DIVIDE
angstrom	1.000 000 x E -10	meters (m)
atmosphere (normal)	1.013 25 x E +2	kilo pascal (kPa)
bar	1.000 000 x E +2	kilo pascal (kPa)
barn	1.000 000 x E -28	meter^2 (m^2)
British thermal unit (thermochemical)	1.054 350 x E +3	joule (J)
calorie (thermochemical)	4.184 000	joule (J)
cal (thermochemical/cm^2)	4.184 000 x E -2	mega joule/m^2 (MJ/m^2)
curie	3.700 000 x E +1	*giga bacquerel (GBq)
degree (angle)	1.745 329 x E -2	radian (rad)
degree Fahrenheit	$t_k = (t^of + 459.67)/1.8$	degree kelvin (K)
electron volt	1.602 19 x E -19	joule (J)
erg	1.000 000 x E -7	joule (J)
erg/second	1.000 000 x E -7	watt (W)
foot	3.048 000 x E -1	meter (m)
foot-pound-force	1.355 818	joule (J)
gallon (U.S. liquid)	3.785 412 x E -3	meter^3 (m^3)
inch	2.540 000 x E -2	meter (m)
jerk	1.000 000 x E +9	joule (J)
joule/kilogram (J/kg) radiation dose absorbed	1.000 000	Gray (Gy)
kilotons	4.183	terajoules
kip (1000 lbf)	4.448 222 x E +3	newton (N)
kip/inch^2 (ksi)	6.894 757 x E +3	kilo pascal (kPa)
ktap	1.000 000 x E +2	newton-second/m^2 ($N-s/m^2$)
micron	1.000 000 x E -6	meter (m)
mil	2.540 000 x E -5	meter (m)
mile (international)	1.609 344 x E +3	meter (m)
ounce	2.834 952 x E -2	kilogram (kg)
pound-force (lbs avoirdupois)	4.448 222	newton (N)
pound-force inch	1.129 848 x E -1	newton-meter (N-m)
pound-force/inch	1.751 268 x E +2	newton/meter (N/m)
pound-force/foot^2	4.788 026 x E -2	kilo pascal (kPa)
pound-force/inch^2 (psi)	6.894 757	kilo pascal (kPa)
pound-mass (lbm avoirdupois)	4.535 924 x E -1	kilogram (kg)
pound-mass-foot^2 (moment of inertia)	4.214 011 x E -2	kilogram-meter^2 ($kg-m^2$)
pound-mass/foot^3	1.601 846 x E +1	kilogram-meter^3 (kg/m^3)
rad (radiation dose absorbed)	1.000 000 x E -2	**Gray (Gy)
roentgen	2.579 760 x E -4	coulomb/kilogram (C/kg)
shake	1.000 000 x E -8	second (s)
slug	1.459 390 x E +1	kilogram (kg)
torr (mm Hg, 0^0 C)	1.333 22 x E -1	kilo pascal (kPa)

*The bacquerel (Bq) is the SI unit of radioactivity; 1 Bq = 1 event/s.

**The Gray (GY) is the SI unit of absorbed radiation.

FINAL REPORT: 07/16/2009 - 08/31/2014

Grant Number: HDTRA1-09-1-0061

Project Title: **Combating Weapons of Mass Destruction: Models, Complexity, and Algorithms in Complex Dynamic and Evolving Networks**

Dr. My T. Thai (Principal Investigator)
Associate Professor
Department of Computer and Information Science and Engineering
University of Florida
Gainesville, FL 32611
Phone: (352) 328-9085
Fax: (352) 392-1220
E-mail: mythai@cise.ufl.edu

Contents

1 Objectives and Relevance

The goal of this project is to address novel quantitative challenges arising in network vulnerability assessment and defense measurement in the face of cascading failures and large-scale attacks such as WMD. More specifically, we aim to accomplish the following three primary goals:

1. **Critical Node Detection.** We investigate the network vulnerability under multiple attacks with different level of disruptions. That is, we aim to identify the most critical subsets of elements (such as nodes and/or edges) whose simultaneous removal maximizes the disruptive effect on the network in term of connectivity. The attacks considered in this study target on both nodes and edges simultaneously. For the networks, we first study static, then dynamic, and modeled them into evolving networks, and finally considered a system consisting of two interdependent networks. The study helps us reveal the most critical location that we need to protect (defense strategies) or attack to break down adversarial networks (attack strategies).

2. **Network Structural Interdependency and Vulnerability Assessment.** Since understanding the interdependency of network structures can reveal the behavior of vulnerability propagation, we propose to investigate the network interdependencies based on their underlying topology focused on the inter- and intra-dependencies between network components and develop a theoretical framework characterizing these interdependences, which has not been provided in the literature. To achieve these goals, we introduce several new models based on an observation that most complex networks exhibit a network modular property, that is nodes within a network module are more densely connected among each other than with the rest of the network, sometimes referred as community structure. A network module may represent a functional group, a component, or an entity within the network system and the correlation among modules can model and describe the interdependency between network components. Along this direction, we aim to investigate the strength of community structures, how to break them, and how community structures evolved in order to predict responses of network components to WMD attacks.

3. **Impact Analysis of the Power-Law Degree Distribution on Network Vulnerability.** As many real-word complex networks such as the Internet, WWW, communication networks, and social networks, have the degrees that follow the power-law distribution, we investigate how this special property will impact on the network vulnerability and its complexity. In particular, we aim to (i) provide a theoretical framework, analyzing the vulnerability of power-law networks under different attacks, (ii) provide near-optimal algorithms to solve many NP-complete problems on power-law networks, and (iii) investigate the hardness complexity of many optimization problems on power-law networks.

Relevance. The study offers the first mathematical study on network vulnerability and defense considering the most realistic scenarios where attacks are dynamic and spreading, and failures are cascaded across several interdependent networks within a complex system. Therefore, it lays a foundation in understanding the fundamental properties that contribute to the network robustness under WMD attacks, and thus, advances the state-of-the-art in modern complex network theory and multi-stage optimization algorithms.

Some of their applications in the area of defense are as follows:

- The applications include the protection of moving military units in which the dynamic network of moving vehicles is considered. The proposal helps to design a more robust strategic network in a WMD stressor environment. This also indicates that the proposed problems can be used to study the planning paths of unmanned vehicles, with wireless units, in order to ensure communication between them. It is also applicable in ad-hoc networks.
- Other applications include emergency responses in the event of failures in transportation networks. The study will help plan the allocation of resources during the evacuation, re-establish critical routes in the aftermath of a disaster, and predict the network responses.
- The study also finds applications in critical infrastructures such as communication networks, wireless networks, transportation networks, and power systems.

The findings also have major impact in the field of networks science and complex networks as many problems studied are on the complex networks and social networks. In addition, the solutions obtained from this project can be used in different field such as vaccination and virus contamination. The results obtained are very basic and have the solid impact on the base of knowledge. For example, the study of network structure is very important to understand any behavior on top of that networks. The study of hardness complexity is important to design algorithms for the problems. The project crosses several research areas such as approximation algorithm, optimization, and control theory, thus it has a profound impact on these fields as well.

2 Major Accomplishments

We have obtained the following findings, corresponding to the above three primary goals.

2.1 Critical Node Detection

We have modeled two main optimization problems as follows:

Definition 1 *β-Vertex (Edge) Disruptor (β-VD (ED)): Given a graph $G = (V, E)$ and $0 \leq \beta \leq 1$, find a subset of vertices (edges) S with minimum size so that the pairwise connectivity of $G[V \setminus S]$ ($G = (V, E \setminus S)$) is at most $\binom{\beta}{n2}$*

Definition 2 *k-CND (k-CED): Given a graph $G = (V, E)$ and a positive integer $k \leq |V|$, find a subset $S \subseteq V$ ($S \subseteq E$) so as to minimize the pairwise connectivity of $G[V \setminus S]$ ($G = (V, E \setminus S)$)*

Relevance. We assess network vulnerability from two different perspectives, namely attack and defense. For example, from an attack point of view, identifying and destroying these critical nodes simultaneously will help seek a maximum destruction in terms of maximum network fragmentation. For example, we can apply this approach to destroy, i.e. arrest, a small number of individuals in an adversarial social network (e.g. terrorist network) in order to maximally disrupt the networks ability to deploy a coordinated attack. From a defensive view, we try to identify the nodes and edges that are considered vital and need to be secured. In a broad sense, we try to minimize the damages in case of a defense or maximize damages in case of an attack, with the available resources. The proposed methods also allow us to accommodate dynamic and evolving networks, which are much

more crucial because dynamics are the key factors that need to be explicitly considered in military settings.

The β-VD (ED) allows us to break the networks to a required extent with a minimum cost while the k-CND tries to maximize the damage with a given cost. Furthermore, different value of β allows us to disrupt the network at different levels. The metric pairwise connectivity provides a global damage view to the networks, instead of a local measurement, an existing metrics used in the literature.

Summary of Findings

1. Provided the best approximation algorithm, $O(\sqrt{\log n})$ ratio for β-VD and β-ED.
2. Provided new mathematical programming approach to find an exact solutions for NP-complete problems with an instance up to a thousand nodes. Current available method (in the literature) is only able to solve upto 100 nodes.
3. Proved the spectral bound for vulnerability assessment in large-scale networks. The result helps us to determine the minimum cost needed to attack (or protect) a given network to reduce the network connectivity under some threshold
4. Provided approximation algorithms in the case of dynamic networks, interdependent networks, in the presence of cascading failures
5. Provided effective defense strategies to spreading and dynamic attacks.
6. Provided models and solutions to strengthen the network modules so that they will be less sensitive to changes, thereby improving network robustness with minimum additional costs.

2.1.1 Detailed Results for Static Networks

First of all, we investigated the above 2 problems where input G is static. We can see G as a network snapshot at time t, and thus it is static. This study lays a foundation to solve the problems in dynamic and evolving networks as shown later.

We have shown that k-CND and β-ED are NP-complete whereas β-VD is MaxSNP-hard. Furthermore, we proved that k-CND is still NP-complete in Unit Disk Graphs and Power-Law graphs. We have proposed two pseudo-approximation algorithm for β-ED and β-VD with the ratio of $O(\log^{1.5} n)$ and $O(\log n \log\log n)$ respectively where n is the size of an input. Via experimental results, we have also shown that the new model is a better way to assess the network vulnerability. More details can be found in papers [47, 52, 54]. We have included here the pseudo-codes as shown in Algorithm β-edge Disruptor and β-vertex Disruptor.

Exact Solutions and Lower Bound. We provided the first exaction solution via mathematical programming for β-vertex disruptor, raising the size of the largest instance solved from a few dozen to several hundreds. This technique is very basic and can be applied for several different problems, not only to the β-vertex disruptor. This finding goes beyond what we have proposed. The result is published in [50].

Algorithm β-edge Disruptor

Input: Uniform directed graph $G = (V, E)$ and $0 \leq \beta < \beta' < 1$

Output: A β'-edge disruptor of G.

/* Construct the decomposition tree */

$c \leftarrow 1 - \sqrt{\frac{\beta}{\beta'}}$.

$T(V_T, E_T) \leftarrow (\{t_0\}, \phi)$, $V(t_0) \leftarrow V(G)$, $l(t_0) = 1$

while $\exists$ unvisited t_i with $|V(t_i)| \geq 2$ **do**

  Mark t_i visited, create new child nodes t_{i1}, t_{i2} of t_i.

  $l(t_{i1}), l(t_{i2}) \leftarrow l(t_i) + 1$.

  $V_T \leftarrow V_T \cup \{t_{i1}, t_{i2}\}$

  $E_T \leftarrow E_T \cup \{(t_i, t_{i1}), (t_i, t_{i2})\}$

  Separate $G[V(t_i)]$ into two using directed c-balanced cut.

  Assign two obtained partitions to $V(t_{i1}), V(t_{i2})$

  $cost(t_i) \leftarrow$ The cost of the balanced cut

end while

/* Find the minimum cost G-partitionable */

for $t_i \in T$ in reversed BFS order from root node t_0 **do**

  for $p \leftarrow 0$ **to** $\beta' \binom{n}{2}$

    if $\mathcal{P}(G[V(t_i)]) \leq p$ **then**

      $cost(t_i, p) \leftarrow 0$

    else

      $cost(t_i, p) \leftarrow \min\{cost(t_{i1}, p_1) +$

        $cost(t_{i2}, p_2) + cost(t_i) \mid p_1 + p_2 = p\}$

Find F with $\mathcal{P}(F) = \min\{cost(t_0, p) \mid p \leq \beta' \binom{n}{2}\}$

Return union of cuts used at $\mathcal{A}(F)$ during tree construction

Algorithm β'-vertex disruptor

Input: Directed graph $G = (V, E)$ and fixed $0 < \beta' < 1$.

Output: A β'-vertex disruptor of G

1. $G'(V', E') \leftarrow (\phi, \phi)$
2. $\forall v \in V : V' \leftarrow V' \cup \{v^+, v^-\}$
3. $\forall v \in V : E' \leftarrow E' \cup \{(v^- \rightarrow v^+)\}, c(v^-, v^+) \leftarrow 1$
4. $\forall (u \rightarrow v) \in E : E' \leftarrow E' \cup \{u^+ \rightarrow v^-\}, c(u^+, v^-) \leftarrow \infty$
5. $\underline{\beta} \leftarrow 0, \overline{\beta} \leftarrow 1$
6. $D_V \leftarrow V(G)$
7. **while** $(\overline{\beta} - \underline{\beta} > \epsilon)$ **do**
8. $\quad \tilde{\beta} \leftarrow \lfloor \frac{\underline{\beta} + \overline{\beta}}{2\epsilon} \rfloor \times \epsilon$
9. $\quad$ Find $D_e \subset E'$ to separate G' into strongly connected components of sizes at most $\tilde{\beta}|V'|$ 10. $\quad D_v \leftarrow \{v \in V(G) \mid (v^+ \rightarrow v^-) \in D_e\}$
11. $\quad$ **if** $\mathcal{P}(G[V \backslash D_v]) \leq \beta \binom{n}{2}$ **then**
12. $\quad\quad \underline{\beta} = \tilde{\beta}$
13. $\quad\quad$ Remove nodes from D_v as long as $\mathcal{P}(G[V \backslash D_v]) \leq \beta \binom{n}{2}$
14. $\quad\quad$ **if** $|D_V| > |D_v|$ **then** $D_V = D_v$
15. $\quad$ **else**
16. $\quad\quad \overline{\beta} = \tilde{\beta}$

18.**end while**

19. Return D_V

We have also proven the spectral bound for link vulnerability assessment in large-scale networks. The result helps us to determine the minimum cost needed to attack (or protect) a given network to reduce the network connectivity under some threshold. Two major findings along this direction are: (1) We introduced a new spectral lower-bound for the β-edge disruptor problem in form of an eigenvalue optimization problems. At the same time, we enriched the literature on lower-bound techniques. (2) We presented two efficient methods to compute the proposed lower-bound: a) the Lagrange multiplier method and b) the dynamic programming algorithm. Moreover, the Lagrange multiplier method can derive the lower-bound with only a small number of smallest eigenvalues. This is important for large networks where computing the whole network spectrum is both time and memory consuming. The result has been published in [9, 29].

Generalization. We next generalized the problems to the case that both edges and vertices are being attacked at the same time as in the following model.

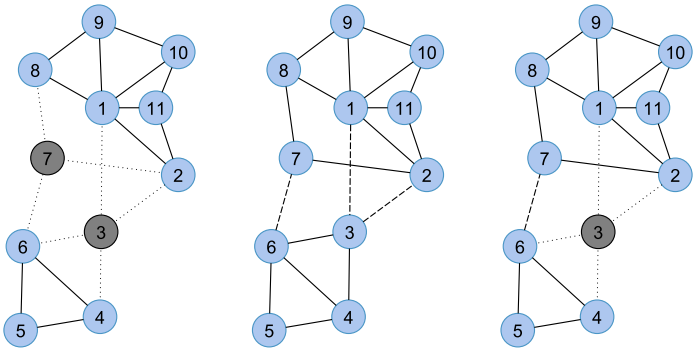
β-disruptor. A β-disruptor is a pair of subsets

$$D_\beta = (V_\beta \subseteq V, E_\beta \subseteq E)$$

that removal from G will make the pairwise connectivity in the residual graph $G' = (V \setminus V_\beta, E \setminus (E_\beta \cup V_\beta \times V_\beta))$ to be at most $\beta\binom{n}{2}$. The *β-disruptor problem* asks for a β-disruptor with the minimum total cost

$$c(D_\beta) = \sum_{u \in V_\beta} c(u) + \sum_{e \in E_\beta} c(e).$$

We further notice that with the same attack cost, if we are allowed to attack both edges and nodes, instead of either edges or nodes separately, a network can be destroyed to a larger extent.



(a) Node attack (b) Link attack (c) Link-node attack

Figure 1: Minimum cost solutions to reduce 50% of the connectivity assuming links have cost 2 and nodes have cost 3 **a.** node only & **b.** link only **c.** joint nodes & links. The minimum cost is 6 if attacking only nodes or only links, and is 5 if both links and nodes are targeted. Thus, it is insufficient to study node and link attacks separately.

Fig. 1 also illustrates a fundamental shortcoming of existing work: the ability to assess network vulnerability under *joint node and link attacks*. The three sub-figures show the minimum cost attack strategies to reduce $\beta = 50\%$ pairwise connectivity, assuming each link has cost 2 and each

node has cost 3. While the minimum costs for both node-attack (Fig. 1a) and link-attack (Fig. 1b) are 6, the minimum cost for node-link attacks (node 3 and link $(6, 7)$) (Fig. 1c) is only 5. Thus, it is insufficient to assess link vulnerability and node vulnerability separately when both links and nodes in the network can be targeted. To make matters worse, assume node 3 and link $(6, 7)$ have the same cost $\epsilon > 0$, the minimum costs for node, link, and node-link attacks will be $3 + \epsilon, 4 + \epsilon$, and 2ϵ, respectively. As the ratios $(3+\epsilon)/(2\epsilon)$ and $(4+\epsilon)/(2\epsilon)$ go unbounded, the existing methods can seriously misjudge the network vulnerability.

To address this shortcoming, we studied the effect of *joint node and link attacks* in term of connectivity. We introduced a new problem, called β-disruptor, that finds a minimum cost set of *nodes and links* whose removal degrades the pairwise connectivity to a great extent (a fraction β). The β-disruptor problem aims to provide a more comprehensive assessment on network vulnerability. It generalizes both the β-vertex disruptor and the β-edge disruptor problems. Along this direction, we proposed an $O(\sqrt{\log n})$-pseudo approximation algorithm for this β-disruptor problem. The main challenge is to address a right cut on directed graphs as the sparsest cut is no longer suitable. The results is shown in [7]. This result also improves our previous results of $O(\log^{1.5} n)$ for β-ED and $O(\log n \log \log n)$ for β-VD.

Variants. We further observed that even before the network beings fragmenting into pieces, its Quality of Service (QoS) may already drop to an intolerant low level, and the network can no longer provide services. To this end, we presented a novel QoS-aware vulnerability assessment framework, called QoS-Critical Vertices (QoSCV) / QoS-Critical Edges(QoSCE) as follows.

Definition 3 *QoS-Critical Vertices (QoSCV) / QoS-Critical Edges(QoSCE): Given a directed graph $G(V, E, s, t)$ with m-dim edge weight vector $(u, v) \in E$: $(w_1(u, v), w_2(u, v), \cdots, w_m(u, v))$. The weight vector for each $s - t$ path P is defined as $(w_1(P), w_2(P), \cdots, w_m(P))$ where $w_i(P) = \sum_{(u,v)\in P} w_i(u, v)$ for all $i \in [1, \cdots, m]$. Given a constraint threshold vector $(c_1, c_2, \cdots, c_m)$ with corresponding credit vector $(\lambda_1, \lambda_2, \cdots, \lambda_m)$, an $s - t$ path P* satisfies *the i^{th} constraint (denoted as $p \propto i$)* iff $w_i(P) \leq c_i$, *and an* SAT *score $\phi(P)$ is defined as: $\phi(P) = \sum_{j:P\propto i} \lambda_j$. The* SAT *score for the graph G is $\phi(G) = \max_{P\in G} \phi(P)$, i.e. the maximum score among all s-t paths. The QoSCV/QoSCE problem is to find a minimum set S of edges/vertices such that $\phi(G \setminus S) \leq \rho$ for a given score threshold ρ.*

We have provided an exact algorithm in case of small m and a heuristic in case of arbitrary m. We include here the psedo-code as shown in Algorithms 1 and 2. All the proofs and experimental results can be found in [56].

2.1.2 Detailed Results for Dynamic and Evolving Networks

We next investigated critical node identification in dynamic networks by two approaches: (i) using the time-series snapshot to represent the data, and (ii) using the probabilistic graphs.

For the time-series snapshot, given a $G_1, G_2, ..., G_t$ representing the network at time $t_1, t_2, ..., t_t$, find the set of critical nodes at time t_i based on the solution of time t_{i-1}. Thus the solution was adaptive from previous time step, rather than computing it from the scratch. Therefore, the algorithm is able to handle the dynamics, thus providing a better tool to adaptively identify a set of critical nodes during a sequence of attacks, or during the recovery process. We have provided

Algorithm 1: Exact Algorithm MFMCSP

Input: directed graph $G = (V, E)$, constraint set $M = \{c_1, \cdots, c_m\}$, credit vector $(\lambda_1, \lambda_2, \cdots, \lambda_m)$, satisfactory score threshold ρ;
Output: solution set of edge of QoSCE.
$S \leftarrow$ all the **minimal** combinations ss of M with $\sum_{c_i \in ss} \lambda_i > \rho$;
for each edge $(i, j) \in E$ **do**
    Set $f(i, j) = f(j, i) = 0$;
    Set $c_f(i, j) = 1$ and $c_f(j, i) = 0$.
end for
while $S \neq \emptyset$ **do**
    $ss \leftarrow$ extracted from S;
    while $\exists q \leftarrow$ the shortest path satisfying all the constraints in ss **do**
        for each edge $(u, v) \in q$ **do**
            $c_f(q) = \min\{c_f(u, v) : (u, v) \in q\}$;
            $f(u, v) = f(u, v) + c_f(q)$; $f(v, u) = -f(u, v)$;
            $c_f(u, v) = c(u, v) - f(u, v)$; $c_f(v, u) = c(v, u) - f(v, u)$;
        end for
    end while
end while
all the vertices reachable from s on the residual network induces a cut $\mathcal{T}$.
Return $\mathcal{T}$.

Algorithm 2: SDOP

Input: directed graph $G = (V, E)$, constant ρ;
Output: a set D of edges to be removed.
Set $D \leftarrow \emptyset$;
while SAT_TEST(G) == YES **do**
    find all m single metric shortest paths $\{p_1, \cdots, p_m\}$
    for all edges $e \in E$ **do**
        Find the one appears in the maximum number of such path.;
    end for
    $D \leftarrow D \cup \{e\}$; $E \leftarrow E \setminus \{e\}$;
end while
Return D.

several fitness functions for each case of changes in a network, including node insertion/removal and link insertion/removal. Based on these fitness functions, we provided two adaptive algorithms, one for the critical node identification, and another for the critical link identification. More details can be found in [30, 31].

For the probabilistic network approach, we modeled a dynamic network using the probabilistic network model which allows us to incorporate the network uncertainty (such as links and nodes disappear and reappear over time) and its prediction. Assessing network vulnerability in probabilistic networks introduces a major challenging problem, that is, to compute the expected pairwise connectivity (EPC) in a network. This problem is related to a famous open problem in network reachability. We have shown that computing EPC is #P-complete and presented several sampling methods to compute such a value. Based on this sampling methods, we further developed solutions to find a set of k critical nodes by formulating the problem as a mathematical programming problem and devising two approaches to overcome the difficulty of having an exponential number of constraints in the mathematical formulation.

Formally, we studied the following problem:

k-Probabilistic Critical Nodes Problem (k-pCNP). Given a probabilistic network $\mathcal{G} = (V, E, p)$ and an integer $0 \leq k \leq n$, find a k nodes subset $S \subset V$ that removal minimizes the expected pairwise connectivity (EPC) in the residual network after removing the nodes in S where EPC is defined as follows:

Algorithm 3: SAT_TEST

Input: directed graph $G = (V, E)$, constant ρ;
Output: YES if a satisfactory path probably exists, NO otherwise.
for every edge $e \in E$ **do**
    $\varphi(e) \leftarrow \sum_{i=1}^{m} \frac{w_i^e}{c_i} \lambda_i$;
end for
$p \leftarrow$ shortest s-t path on metric φ;
if $\varphi(p) > \rho$ **then**
    Return NO;
else
    Return YES;
end if

$$\mathsf{EPC}(\mathcal{G}) = \frac{1}{2} \sum_{u,v \in V; u \neq v} \mathsf{REL}_{u,v}(\mathcal{G})$$

where $\mathsf{REL}_{u,v}(\mathcal{G})$ is the probability that v is reachable from u within $\mathcal{G}$.

We have proven that computing this EPC is #P-complete (details of the proof can be found in [13]), let alone finding a set of such k nodes. Since every #P-complete problem either has a *fully polynomial randomized approximation scheme* (FPRAS) or is essentially impossible to approximate, we are interested in (ϵ, δ)-approximations for $\mathsf{EPC}(\mathcal{G})$, i.e., algorithms returning an estimation of $\mathsf{EPC}(\mathcal{G})$ accurate to within a relative error of ϵ factor with probability at least $1 - \delta$. An (ϵ, δ)-approximation is called an FPRAS if its running time is bounded by a polynomial in $1/\epsilon, log(1/\delta)$, and the instance input size. Along this direction, we have developed the following algorithm.

Algorithm (ϵ, δ) **Component Sampling Algorithm to compute** $\mathsf{EPC}(\mathcal{G})$

1. Let $P_E = \sum_{e \in E} p_e$
2. **if** $P_E < \frac{\epsilon}{2} n^{-2}$ **then**
3. return $\mathcal{E}_2 = P_E$.
4. $C_2 \leftarrow 0$.
5. **for** $i = 1$ to $N_2(\epsilon, \delta)$ **do**
    - Select a node $u \in V$ uniformly.
    - Start a Breath-First Search from u. For each encountered edge (v, w), flip a coin of bias p_{vw} to determine its availability.
    - Let S_i be the number of visited nodes.
    - $C_2 = C_2 + (S_i - 1)$.
6. Return $\mathcal{E}_2 = \frac{nC_2}{2N_2}$ as an unbiased estimator of $\mathsf{EPC}(\mathcal{G})$.

The beauty of the above algorithm is that it is proven to be an FPRAS. The proofs can be found in [13]. This fast and accurate computing algorithm can be used in many applications that need to

compute the pairwise connectivity such as the network reachability problem.

We next present our solution for the k-pCNP which is a two-stage stochastic programming, which is formulated as follows:

$$\min_{s\in\{0,1\}^n} \quad \mathbb{E}\left[P(s,x,\xi)\right] \tag{1}$$

$$\text{s. t.} \sum_{i=1}^{n} s_i \leq k \tag{2}$$

$$\text{where } P(s,x,\xi) = \min \sum_{i<j} (1 - x_{ij}) \tag{3}$$

$$\text{s. t. } x_{ij} \leq s_i + s_j + 1 - \xi_{\mathbf{ij}},\ (i,j) \in E, \tag{4}$$

$$x_{ij} + x_{jk} \geq x_{ik}, \quad (\mathbf{i},\mathbf{j}) \in \mathbf{E}, k = 1..n \tag{5}$$

$$x_{ij} = x_{ji}, \quad i,j = 1..n \tag{6}$$

$$s \in \{0,1\}^n, x \in [\mathbf{0},\mathbf{1}]^{n^2} \tag{7}$$

Discretization. To solve the stochastic program numerically, one needs to consider all possible realization $G^l \in \mathcal{S}_\mathcal{G}$ and their probability masses $f_\mathcal{G}(G^l)$. Then the two-stage stochastic program can be written as a (one-level) mixed integer programming, denoted by MIP$_F$:

$$\min \sum_{l=1}^{N} f_\mathcal{G}(G^l) \sum_{i<j} (1 - x_{ij}^l) \tag{8}$$

$$\text{s. t.} \sum_{i=1}^{n} s_i \leq k \tag{9}$$

$$x_{ij}^l \leq s_i + s_j + 1 - \xi_{\mathbf{ij}}^{\mathbf{l}}, (i,j) \in E, l = 1..N \tag{10}$$

$$x_{ij}^l + x_{jk}^l \geq x_{ik}^l, \quad (i,j) \in E, k = 1..n, l = 1..N \tag{11}$$

$$x_{ij}^l = x_{ji}^l, \quad i,j = 1..n, l = 1..N \tag{12}$$

$$s \in \{0,1\}^n, x^l \in [0,1]^{n^2},\ l = 1..N \tag{13}$$

The major challenge in solving this discretized form is that there is an exponential number ($N = 2^{|E|}$) of variables and constraints. Thus, solving MIP$_F$ is intractable even for very small instances of $\mathcal{G}$. To overcome this difficulty, we developed two approximate mathematical programs of substantially smaller sizes: (1) Approximating via the expectation graph and (2) Sample average approximation method which reduces the number of realizations. The two solutions are shown in the following pseudo-codes.

Algorithm Rounding on the Expectation Graph Algorithm (REGA)

1. Obtain an LP relaxation of MIP_E with the relaxed constraints $s \in [0,1]^n$.
2. Initialize the set of selected nodes $D = \emptyset$.
3. Repeat k times the following steps
   - Solve the LP relaxation
   - Select $u = \arg\max \in V \setminus Ds_i$.
   - Add u to D and fix $s_u = 1$
4. Return k critical nodes in D.

Algorithm Sample Ave. Approx. Algorithm (SA3)

Parameter T: the number of sampling

Phase 1: Delayed Constraints

1. Initialize an LP with the objective $\frac{1}{T}\sum_{l=1}^{T}\sum_{i<j}(1 - x_{ij}^l)$ and only the constraints $s \in [0,1]^n, x_{ij}^l \in [0,1]$.
2. **for** $l = 1..T$ **do**
   - Generate the l^{th} sample of $\mathcal{G}$ (adjacency matrix ξ^l).
   - Add the constraints involved x_{ij}^l to the LP.
   - Solve the updated LP.

Phase 2: Iterative rounding

3. Initialize the set of selected nodes $D = \emptyset$.
4. Repeat k times the following steps
   - Select $u = \arg\max_{i \in V \setminus D} s_i$.
   - Add u to D and fix $s_u = 1$
   - Re-solve the LP
5. Return k critical nodes in D.

2.1.3 Detailed Results for Interdependent Networks

We finally investigated the critical node detection problems on interdependent networks. Although there exist some work on the vulnerability assessment of interdependent networks, most of them focus on the artificial models of interdependent networks, i.e., random interdependency between networks, and ignore the detection of top critical nodes in real networks. Therefore, we study the following new optimization problem:

Definition 4 (IPND problem) *Given an integer k and an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, which consists of two networks $G_s = (V_s, E_s)$, $G_c = (V_c, E_c)$ along with their interdependencies E_{sc}. Let $LG_s(T)$ be the size of the largest connected component of G_s after the cascading failures caused by the initial removal of the set of nodes $T \subseteq V_s$ in G_s. The IPND problem asks for a set T of size at most k such that $LG_s(T)$ is minimized.*

We used a well-accepted cascading failure model, which has been validated and applied in many previous works. Initially, there are a few critical nodes that fail in network G_s, which disconnects a set of nodes from the largest connected component of G_s. Due to the interdependency of two networks, all the nodes in G_c that connect to failed nodes in G_s are also affected, and therefore stop working. Furthermore, the failures cascade to nodes which are disconnected from the largest connected component in G_c and cause further failures back to G_s. The process continues back and forth between two networks until there are no more failure nodes.

Our major findings are two fold: (1) We showed the $(2 - \epsilon)$-inapproximability of the IPND problem on interdependent networks; and (2) We provided the greedy framework with various centralities to solve the IPND problem. We validated the performance of our solutions on a wide range of interdependent networks with different scales, topologies, and interdependencies. The proposed centrality function on interdependent networks is very important since it can be served as a basic function for many research works in the future which requires a centrality function.

In a maximum cascading (Max-Cas) algorithm, we iteratively select a node u that leads to the most number of new failed nodes, i.e., the maximum marginal gain to the current set of attacked nodes T. When a new node u fails, it results in a chain of cascading failures. The number of new failed nodes, referred to as *cascading impact number*, can be computed by simulating the cascading failures with the initial set $T \cup \{u\}$ on the interdependent system $\mathfrak{I}$. However, the simulation of cascading failures is time-consuming due to its calculation of cascading failures between two networks. Each step in the cascading requires to identify the largest connected component of each network.

To this end, we further improved the running time of our algorithm by reducing the number of simulations. The idea is to check potential nodes whose removal creates at least one more failed node in the same network due to the cascading failures. That is, this node (or its coupled node) disconnects the network to which it belongs, i.e., it (its coupled node) is an articulation node of G_s (or G_c), which is defined as any vertex whose removal increases the number of connected components in G_s (or G_c). The reason is illustrated in the following lemma.

Lemma 1 *Given an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, removing a node $u \in V_s$ from the system causes at least one more node fail due to the cascading failure iff u (or its coupled node $v \in V_c$) is an articulation node in G_s (or G_c).*

According to this property, the proposed algorithm first identifies all articulation nodes in both residual networks using the Hopcroft and Tarjan's algorithm. Note that this algorithm runs in linear time on undirected graphs, which is faster than one simulation of cascading failures. Thus, the run time of each iteration is significantly improved especially when the number of articulation nodes is small. Denote $Max - Cas(G_s, T, \{u\})$ as the impact number of u, Algorithm 4 describes the details to detect critical nodes. In Algorithm 4, since it takes $O(n)$ time to compute the cascading impact number for each node and at most $|A| < n$ articulation nodes will be evaluated, the run

Algorithm 4: Max-Cas Greedy Algorithm

Input: Interdependent system $\Im(G_s, G_c, E_{sc})$, an integer k
Output: Set of k critical nodes in $T \in V_s$
$T \leftarrow \emptyset$
for $i = 1$ to k **do**
  $A_s, A_c \leftarrow$ set of articulation nodes of G_s and $G_c$respectively
  $A \leftarrow \{u \in V_s | u \in A_s \vee ((u, v) \in E_{sc} \wedge v \in A_c)\}$
  if $A \neq \emptyset$ **then**
    $u \leftarrow \text{argmax}_{u \in A}$ Max-Cas$(G_s, T, \{u\})$
    $T \leftarrow T \cup \{u\}$
  else
    $u \leftarrow$ any node in $V_s \setminus T$
  end if
  Update $\Im[V_s \setminus T]$
end for
Return T

time is $O(kn^2)$ in the worst case. In practice, the actual run time is much less due to the small size of A, which is shown in our experiment.

For the new centrality measure in interdependent networks, this measure is required to capture both the intra-centrality (the centrality of nodes in each networks) and inter-centrality (the centrality formed by the interconnections between two networks). Given an interdependent system $\mathcal{I}(G_s, G_c, E_{sc})$, node $u \in V_s$ is more likely to be critical if its coupled node $v \in G_c$ is critical. Furthermore, when node u is considered as a critical node, its neighbors are also more likely to become important since the failures of these nodes can cause u failure. That said, the criticality of these nodes imply the criticality of their coupled nodes. To capture this complicated relation in interdependent systems, we develop an iterative method to compute the centrality of nodes, called Iterative Interdependent Centrality (IIC). Initially, the centralities of all nodes in G_s are computed by the traditional centrality, e.g., degree centrality, betweenness centrality, etc. After that, these centralities of nodes in G_s are reflected to coupled nodes in G_c and the centralities of nodes in G_c are updated based on the reflected values. The centralities of nodes in G_c continue to be reflected on nodes of G_s and update centralities of these nodes. Two key points of IIC are the updating function and the convergence.

The updating function is defined as follows:

$$\mathcal{C}(u) = \alpha w(u) + (1 - \alpha) \sum_{v:(u,v) \in E} \frac{w(v)}{d_v}$$

where $w(\cdot)$ is the centralities of nodes and the reservation factor α lying in the interval $[0, 1]$. The underlying reason we use centrality-based degree is that a node is usually more critical if most of its neighbors are critical nodes. We can easily modify this function to cope with the weighted graphs. This centrality can be computed based on matrix multiplications and we have proved its convergence in our paper [32], which was published on the IEEE Transactions on Smart Grid.

k Interdependent Networks. In order to solve the above IPND problem in k interdependent networks, we devised an interdependent centrality method. This method is not only used to solve this problem, but also can be applied for other problems which requires centrality. Note that many centrality methods have been developed for individual networks. But none has been studied for the k interdependent networks. The two major challenges of this problem are to measure the centrality

of a node with respect to its own networks, and then to other networks. Furthermore, the iteratively updating must converge. To this end, we have developed the following method.

When two nodes are coupled, the failure of one node is equivalent to the failure of the other node, thus both of them should have the same centrality value. There are two kinds of nodes: (1) nodes have no coupled nodes in other networks and (2) nodes have at least one coupled node in other networks. The centrality of the former type of node should depend only on the network structure. On the other hand, the centrality of the later type of node should depend on neighbor nodes in multiple networks. The key idea to design the new centrality is to decompose the centrality of the coupled nodes into different component corresponding to different networks.

If u belongs to only network G^i, then:

$$C(u) = \frac{1}{2}C(u) + \frac{1}{2}\sum_{v:(u,v)\in E^i}\frac{C(v)}{d_v}$$

If $u^{i_1}, \ldots, u^{i_p}$ are p coupled nodes of networks $G^{i_1}, \ldots, G^{i_p}$, then the centrality of these nodes will be:

$$C(u^{i_1}) = \ldots = C(u^{i_p}) = \frac{1}{\sum_{t=1}^{p}\beta_{i_t}}\sum_{t=1}^{p}\beta_{i_t}\left(\frac{1}{2}C(u^{i_t}) + \frac{1}{2}\sum_{v:(u^{i_t},v)\in E^{i_t}}\frac{C(v)}{d_v}\right)$$

$$= \frac{1}{\sum_{t=1}^{p}\beta_{i_t}}\sum_{t=1}^{p}(\beta_{i_t}\alpha C(u) + (1-\alpha)\sum_{v:(u_{i_t},v)\in E^{i_t}}\frac{C(v)}{d_v})$$

The parameter β_{i_t} shows the fragility of the network G^{i_t}. If network G^{i_t} is very fragile e.g. very sparse, the value of β_{i_t} is large.

The remaining key point for this method is to prove its convergence. We proved it via the following steps. (We include the sketch of proofs here as the paper is being written and thus is not available at this time)

Definition 5 *The associated graph of a nonnegative square matrix A is a directed graph G_A of n vertices, where n is the size of A. G has edge from vertex i to vertex j when $A_{ij} > 0$.*

Lemma 2 *The nonnegative matrix A is irreducible if and only if its associated graph G_A is strongly connected.*

Lemma 3 *The nonnegative matrix A is aperiodic if and only if the greatest common divisor of the lengths of the closed directed paths in its associated graph G_A is 1.*

Theorem 1 (Perron-Frobenius theorem) *If the nonnegative matrix A is irreducible and aperiodic, then there exists a positive real number r such that r is an eigenvalue of A and any other eigenvalue λ is strictly smaller than r in absolute value, $|\lambda| < r$.*

Theorem 2 (Perron-Frobenius theorem) *If the nonnegative matrix A is irreducible and aperiodic, then there exists a positive real number r such that r is an eigenvalue of A and any other eigenvalue λ is strictly smaller than r in absolute value, $|\lambda| < r$.*

$$x^t = \frac{M^k M^{k-1} \dots M^1 x^{t-1}}{|M^k M^{k-1} \dots M^1 x^{t-1}|} = \frac{M x^{t-1}}{|M x^{t-1}|}$$

The metric will converge if $M = M^k M^{k-1} \dots M^1$

$$M^i_{uv} = \begin{cases} \alpha & \text{if } u = v \\ (1-\alpha)/d^i_v & \text{if } (u,v) \in E^i \\ 0 & \text{otherwise} \end{cases}$$

The uniqueness of the centrality vector x^t will converge to a unique vector x regardless of the initial vector x^0.

Let $v_1, v_2, \dots, v_n$ be the basis of eigenvectors of M, we have:

$$M v_i = \lambda_i v_i$$

x^0 can be represented by basis as: $x^0 = \sum_i c_i v_i$

Then:

$$M^t x^0 = \sum_i c_i M^t v_i = \sum_i c_i \lambda_i^t v_i$$

$$\frac{M^t x^0}{c_1 \lambda_1^t} = v_1 + \frac{1}{c_1} \sum_{i>1} \left(\frac{\lambda_i}{\lambda_1}\right)^t c_i v_i$$

$$x[u] = \alpha x[u] + (1-\alpha) \sum_{v:(u,v)\in E} \frac{x[v]}{d_v}$$

2.1.4 Detailed Results for Cascading Failures

We re-investigated the above problems in the presence of cascading. Not only can the failures be cascaded, but the attack itself can also be propagated. For example, chemical agents can spread through a network (e.g., water), viruses spreading over computer networks, or even deceptive messages influence over online social networks. Thus in this task, we further exploited the network vulnerability considering these type of attacks, thereby providing a much more effective defense strategy. We focused on finding the minimum number of nodes such that if these nodes are attacked, the attack can spread into the whole network and create the greatest damage. Therefore, these nodes are the most vulnerable to this type of attack. As different attacks have their own propagation model, we investigated this problem on various propagation models. In addition, the attack cannot spread infinitely, thus we set the time constraint on the spreading. Along this direction, we formulated the following new optimization problems and studied its inapproximability along with its approximation algorithms.

Definition 6 (Critical Spreading Nodes Identification (CSNI).) *Given a complex system S, a latency bound d, and a propagation model P, find a minimum subset nodes N such that by launching attacks at N initially, all other nodes in S will be "infected" within at most d hops under P.*

First, we considered P as a linear threshold model. In this model, a node v is infected (or refer as activated) iff there are more than $\rho_v d(v)$ neighbors of v active, where $d(v)$ denotes the degree of node v and ρ_v is some input threshold. Once v is active, v can continue in the process of spreading the influence and infect other nodes. The spread will continue after d hops. In [22, 35], we have shown that when $d = 1$, CSNI cannot be approximated within $\ln \Delta - O(\ln \ln \Delta)$ unless $P = NP$ where Δ is the maximum degree of G. We also showed that it cannot be approximated within $\ln B - O(\ln \ln B)$ with degrees bounded by B, and cannot be within $(1/2 - \epsilon) \ln n$, unless $NP \in DTIME(n^{o(\log \log n)})$. When $1 < d < 4$, the problem cannot be approximated within $O(\ln l)$ and for $d > 4$, it cannot be approximated within $2^{\log^{1-\epsilon} n}$. All detailed proofs can be found in [22, 35].

For $d = 1$, we developed a tight approximation algorithm with ratio $H((1 + \rho)\Delta)$ where $H(n)$ is the Harmonic function. In general case where $d < 4$, we devised a greedy algorithm with a tight $\log n$ ratio in the power-law graphs. The algorithm can be run in a very large network, consisting of millions nodes and edges [22]. The algorithm can be modified to cope with a case that we just want to infect a certain portion of the network, not as the whole.

Effective Defense Strategies to Dynamic Attacks with Cascading Failures. We investigated the serial attack points in order to maximize the number of failed nodes after the cascading failures. These attack points are considered as the most vulnerable nodes which need to be protected during the attacks. We provided theoretical analysis, including inapproximalblity results, and algorithmic solutions for this problem.

Due to the cascading failures, the failures of a small set of nodes S can result in a catastrophic number of failed nodes. Therefore, these nodes in set S become the most critical nodes. Additionally, the order of these nodes to be destroyed can lead to different outcomes. In this study, we considered a scenario in which attacks can be launched after another, that being said, the nodes in S are destroyed in a certain order to obtain the maximum malfunction of the network. Furthermore, nodes are failed in the cascading manner due to the load redistribution of failed nodes, called the Load Redistribution model (LR-model). In this model, a set of nodes S are failed initially, then the failures are propagated to other nodes in time steps. When node u fails, its load is redistributed to its neighbors and each alive neighbor will received an additional load which is proportional to its weight. Precisely, each neighbor v of u will receive the following additional load:

$$\Delta L(v) = L(u) \times \frac{w(v)}{\sum_{l \in N_u^+} w(l)}$$

Due to the load redistribution, the load of some nodes are exceeding their capacities, hence fail in the next time step. The process of load redistribution and node failing will stop when there are no more failed nodes. The set of failed nodes caused by the initial failure of S is denoted by $F(S)$.

In particular, given an order set $S = \{s_1, s_2, \ldots, s_k\}$, the set of failed nodes after s_i is attacked is $F_i(S) = F(F_{i-1}(S) \cup \{s_i\})$. Denote $F^+(S)$ as $F_k(S)$, the set of failed nodes when nodes in S is attacked serially. We formally define the problem as follows.

Definition 7 (Cascading Critical Node Problem (Cas-CNP)) *Given a network $G = (V, E)$ and an integer k, the problem asks to find a ordered subset $S \subseteq V$ of size $|S| = k$ such that the serial failures of nodes in S maximizes the number of failed nodes $F^+(S)$ under the LR-model.*

Our major findings of this task are two fold: (1) Showed the $O(n^{1-\epsilon})$-inapproximability of theCas-CNP; (2) Provided the cascading potential metric to solve the problem.

In details, our results are described as follows.

Theorem 3 *It is NP-hard to approximate the CasCN problem within ratio of* $O(n^{1-\epsilon})$ *for any constant* $1 > \epsilon > 0$.

We use the gap-introduction reduction to prove the inapproximability of the CasCN problem, using a polynomial time reduction from a restricted variant MIN3SC2 of the Set Cover problem. The proof can be found in [3].

Cascading Potential Metric. In the dynamic attacks (attacks can be launched after the other), we need to consider the co-impact of attacks to obtain a large number of failed nodes. Therefore, we developed the cascading potential of node u, denoted by $\mathcal{C}(u)$ as follows:

$$\mathcal{C}(u) = \frac{|F(\{u\})|}{n} + \frac{\sum_{v \in V - F(\{u\})} \Delta L_u(v)}{\sum_{v \in V - F(\{u\})} (C(v) - L(v))}$$

where $F(\{u\})$ is the set of failed nodes when u fails and $\Delta L_u(v)$ is the additional load that v receives due to the failure of u.

This metric helps to evaluate the importance of nodes wrt different attack scenarios. Based on this metric, we developed the following Algorithm 5 to the Cas-CNP problem.

Algorithm 5: Cascading Potential Algorithm

Input: A network $G = (V, E)$, an integer k and parameter α.
Output: A set S of k attacked nodes.
Compute the centrality of all nodes
Sort nodes in non-increasing order of centrality $\mathcal{C}(u_1) \geq \mathcal{C}(u_2) \geq \ldots \geq \mathcal{C}(u_n)$
Initialize $S \leftarrow \emptyset$
$j \leftarrow 1$
for $i = 1$ to k **do**
    while $u_j \in F^+(S)$ **do**
        $j \leftarrow j + 1$
    end while
    $S \leftarrow S \cup \{u_j\}$
end for
Return S

Cooperative Attacks. The above strategies focus on selecting nodes to maximize the number of new failed nodes and redistributed load. However, if nodes have a high tolerance factor, they can stand still under the additional load redistributed from attacked nodes. As a consequence, the load of many nodes is increased, but there is no or only a few more nodes failed. In this case, the total number of failed nodes at the end of the cascading process is very low. Therefore, we developed another solution to overcome this challenging.

Since a node u will fail when its load passes its capacity, we can use the difference $C(u) - L(u)$ as the health of u. The weaker u is, the more damage it receives under the same amount of attack ΔL. Thus, the preference of attacking u can be defined as:

$$\gamma(u, \Delta L) = \begin{cases} \frac{\alpha \Delta L}{C(u) - L(u)} & \textit{if } \Delta L + L(u) \leq C(u) \\ 1 & \textit{otherwise} \end{cases}$$

where $0 < \alpha < 1$ is a tunable parameter.

The benefit of this attack preference function is that the more wounded a node is, the higher attack preference it has. This property is stated in Lemma 4.

Lemma 4 *For any node u at two points of time, if u is more wounded at the second point, i.e., $L_2(u) > L_1(u)$, then the attack preference under the same attack ΔL is higher at the second point: $\gamma_2(u) \geq \gamma_1(u)$.*

In such a cooperative attack, we defined the efficiency of selecting the next v to be destroyed. If the load redistributed from v to u is $\Delta L(u)$, the efficiency of v in taking down u is:

$$\lambda(v, u) = \gamma(u, \Delta L(u))\sigma(L(u))$$

The overall efficiency of v is total:

$$\lambda(v, u) = \sum_{u \in V \setminus v} \lambda(v, u)$$

The efficiency function has the following properties:

Lemma 5 *Given a fixed load $L(u)$ and attack ΔL with $L(u) + \Delta L \leq C(u)$, the efficiency on u is monotone decreasing and goes to 0 when the capacity $C(u)$ increases and goes to infinity.*

Lemma 6 *Suppose that the capacity $C(u)$ is linear to the load $C(u) = T * L(u)$ with constant factor T. Then, given a fixed attack ΔL with $L(u) + \Delta L \leq C(u)$, the efficiency on u is monotone non-increasing and goes to 0 when the load $L(u)$ increases and goes to infinity.*

Based on the efficiency evaluation, we developed the Cooperating Attack (CA) algorithm as shown Algorithm 6.

Identification of Critical Nodes with Threshold Cascading Failure. In this study, we investigated the vulnerability of networks under simultaneous attacks with cascading failures. More specifically, we identified k most critical subsets of a network whose simultaneous removal will minimize the total pairwise connectivity of the remaining network under the cascading failure. That being said, once a node v has more than ρ_v neighbor nodes failed, v will fail and this failure will be propagated further. We devised an Integer Programming (IP) with sparse metric to obtain the optimal solution for networks with couple thousands of nodes. For larger networks, we proved the inapproximability and designed a near-optimal solution. Furthermore, we developed a centrality method to identify critical nodes in k-interdependent networks.

We focused on finding critical nodes in (i) an individual networks with threshold cascading failure and (ii) k interdependent networks. More specifically, we have investigated the following Cascading Critical Node Detection (CCND) problem:

Algorithm 6: Cooperating Attack (CA)

Input: A network $G = (V, E)$ and an integer k.
Output: A set S of k seed nodes.
Initialize $S \leftarrow \emptyset$
for $i = 1$ to k **do**
    Form G_i as the network after the failure of S in G
    Evaluate the efficiency of all node in G_i
    Select u as the node with highest efficiency
    $S \leftarrow S \cup \{u\}$
end for
Return S

Definition 8 *Given two integers k, d, a vector of fractional numbers with size n where each $\theta_u \in (0, 1)$ and an undirected graph $G = (V, E)$. Let $P(S)$ be total pairwise connectivity of residual graph G after the d-hop cascading failures caused by the initial removal of the set of nodes $S \in V$. The CVND problem asks for k most vulnerable nodes such that $P(S)$ is minimized.*

For the cascading failure, we considered the threshold model in which each node u in the network has a threshold $\theta_u \in [0, 1]$, typically drawn from some probability distribution. Starting with an initial set of failure nodes F_0, the dynamics of failure cascades unfold round by round as follows. The cascading process is deterministically in discrete rounds: in round t, all nodes that failed in round $t-1$ remain failed, and another node v fails if the total number of its failure neighbors is at least θ_u, i.e., $|N(u) \cap F_{t-1}| \geq \theta_u deg(u)$, in which F_{t-1} is the set of failure nodes before round $t-1$.

We first proved it is NP-hard to be approximated into $\Omega\big(\frac{(1+\frac{d}{n^{1-\epsilon}-1})^2(n-k)}{n^\epsilon}\big)$ where n is the size of the network, which makes it unrealistic for one to quickly obtain optimal solutions within the time constraint. To this end, we proposed TRGA, an iterative 2-phase algorithm to effectively solve these problems in a timely manner. In a big picture, TRGA algorithm detects the ultimate failure nodes after cascading failures and traces back to the critical nodes in each iteration, and terminates until k critical nodes are detected. TRGA algorithm also takes into account the local search, constraint pruning and lazy-update techniques in order to further improve its efficiency and shorten its processing time. In addition, we formulated the mathematical programming to achieve its optimal solution and applied a sparse metric technique to reduce the number of constraints. The performance of TRGA algorithm was validated on both real-world and synthetic networks with different topologies.

More specifically, we proved the following theorem:

Theorem 4 *Assuming $P \neq NP$, the CVND problem is NP-hard to be approximated within $\Omega\big(\frac{(1+\frac{d}{n^{1-\epsilon}-1})^2(n-k)}{n^\epsilon}\big)$ for any $\epsilon < 1 - \log_n 2$ on general graphs.*

The detailed proofs can be found in [1].

Based on the above inapproximability result, we provided the following IP with sparse metric technique to handle small networks with size of thousands nodes.

For each node $i \in V$ and all integers $t \in [0, d]$, we define

$$v_i^t = \begin{cases} 1, & \text{if node } i \text{ fails in round } t \\ 0, & \text{otherwise} \end{cases}$$

Note that $v_i^0 = 1$ when node i is a vulnerable node and fails at the beginning. Then, using u_{ij} defined as above, we have the following ILP:

$$\begin{aligned}
\min \quad & \sum_{i,j \in V} u_{ij} \\
\text{s.t.} \quad & v_i^d + v_j^d + u_{ij} \geq 1 && \forall (i,j) \in E \\
& u_{ij} + u_{jh} - u_{hi} \leq 1 && \forall i,j,h \in V \\
& \sum_{i \in V} v_i^0 \leq k \\
& \sum_{j \in N(v_i)} v_j^{t-1} + \theta \cdot deg(v_i) v_i^{t-1} \\
& \geq \theta \cdot deg(v_i) v_i^t && \forall i \in V, \forall 0 \leq t \leq d \\
& v_i^t \geq v_i^{t-1} && \forall 0 \leq t \leq d \\
& \forall i \in V, 0 \leq t \leq d \\
& v_i^t \in \{0,1\} && \forall 0 \leq t \leq d \\
& u_{ij} \in \{0,1\}
\end{aligned}$$

where the objective is to minimize the total pairwise connectivity. The first constraint guarantees that at least one endpoint of a link has to be deleted after d round cascades if its two endpoints are disconnected in the optimal solution. The second constraint imposes the triangular connectivity. That is, if node i and j are connected, node j and h are connected, node i and h have to be connected. The third constraint means that the total pairwise connectivity after d round failure cascades is at most β fraction of all node-pairs. The last two constraints deals with the cascades process and keeps failed nodes to be failure in the following rounds respectively.

For the large-scale networks, we provided a 2-phase TRGA algorithm as shown in Algorithm 7.

2.2 Network Structural Interdependency and Vulnerability Assessment

We continued to investigate the network vulnerability from the network structure perspective. There are several reasons to study this approach: (i) Changing network structures will change the network functions, and thus may break down the system, (ii) The changes or failures occurred in one module can have a profound impact which can consequently lead to the transformation of other modules, thus requiring us to understand the inter- and intra-interdependence within these modules.

We used community structure to approximate the network structure. To accomplish the second goal, we studied the following: (1) Prediction models based on network modular structures to characterize and forecast the interdependent responses of network components in evolving networks with limited data. Including the dynamics and evolution of a network in the analysis of

Algorithm 7: TRGA Algorithm

Input : Network G, Threshold vector θ
Output: The set of k vulnerable nodes S

$k' \leftarrow k$;
$S \leftarrow \emptyset$;
while $|S| < k$ **do**
  $k' \leftarrow k - |S|$;
  $D \leftarrow k'$ largest degree nodes in $G[V \setminus S]$;
  $\mathfrak{P} \leftarrow$ #failed nodes after cascading failures by removing D from $G[V \setminus S]$;
  $U \leftarrow \emptyset$;
  `// Ultimate Failure Nodes Identification`
  while *Pairwise Connectivity* $> \mathfrak{P}$ **do**
    Use Constraint Pruning in [**?**] to solve the LP formulation with $\mathfrak{P}$;
    $\mathfrak{P} \leftarrow \mathfrak{P}-$ disconnected node-pairs after removing u;
    $u \leftarrow$ the node with largest v_i^*;
    $U \leftarrow U \cup \{u\}$;
    $G \leftarrow G[V \setminus \{u\}]$;
  end
  `// Critical Nodes Tracing Back`
  $Q \leftarrow \emptyset$; `// Priority Queue`
  $S' \leftarrow \emptyset$;
  while $\exists$ *one node does not fail* **do**
    if $Q = \emptyset$ **then**
      foreach *node* u **do**
        Calculate the cascading influence after removing u from G;
      end
      Construct Q based on cascading influence of each node;
    end
    else
      $S' \leftarrow S' \cup$ the node in Q with max priority;
      Update cascading influence caused by removing this node;
    end
  end
  if $|S'| > k'$ **then**
    $S \leftarrow S \cup k'$ largest degree nodes in $G[V \setminus S]$;
  end
  else
    $S \leftarrow S \cup S'$;
  end
end
`// Local Search`
$S^* \leftarrow S$;
foreach *node* $u \in S$ **do**
  Swapping(u);
end
$S \leftarrow S^*$;
return S;

inherent modules is a new task that has not yet been well investigated. The study of such evolution requires computing modules of the network at different time instances. However, identifying network modules in each state of the network from scratch may result in prohibitive computational costs, particularly in the case of highly dynamic networks. In addition, it may be infeasible in the case of limited topological data. In this regard, the study requires us to solve many interesting problems such as: (1) How to devise new measures and methods for computing network modules which are robust to changes, (2) How to easily update the modules once the changes occur without re-calculating them from scratch, (3) How sensitive the community structure is with respect to the failues of nodes and edges.

Relevance. This study helps us understand the interdependencies of network components and provide a robust solution for many applications which are sensitive to the structure of network communities. This study also addresses the improvement of network robustness with minimum additional costs so that the network modules are less sensitive to changes, thus providing a cost-effective protection scheme.

The knowledge about this crucial vulnerability of network community structure is not only helping to understand the network interdependencies but also of considerable usages, particularly having many applications for social-aware methods in mobile ad-hoc and online social networks (OSNs). For instance, since social-based forwarding and routing strategies in Delay Tolerant Networks rely heavily on the highest ranking node in each community to forward the message, the awareness of this vulnerability can help to design either a routing algorithm that do not overload those crucial nodes, or to design an effective backup plan when some of them may fail at the same time.

Summary of Findings.

- Provided the first constant approximation algorithm to find community structure in power-law networks and trees with performance guarantee. This is important first step to predict the changes of community structure.
- Provided the first adaptive approximation algorithms for both disjoint and overlapping community structure. The adaptive solutions help us update the community structure during the changes in a very short amount of time.
- Assessed the network structure vulnerability during attacks, both at edges and nodes. Showed that community structure is not as strong as we think.

2.2.1 Detailed Results for Identifying Community Structures

Consider a network represented as an undirected graph $G = (V, E)$ consisting of $n = |V|$ vertices and $m = |E|$ edges. The *adjacency matrix* of G is denoted by $\boldsymbol{A} = (A_{ij})$, where A_{ij} is the weight of edge (i, j) and $A_{ij} = 0$ if $(i, j) \notin E$. We also denote the (weighted) degree of vertex i, the total weights of edges incident at i, by $\mathsf{deg}(i)$ or, in short, d_i.

Community structure (CS) is a division of the vertices in V into a collection of disjoint subsets of vertices $\mathcal{C} = \{C_1, C_2, \ldots, C_l\}$ (with **unspecified** l) where $\bigcup_{i=1}^{l} C_i = V$. Each subset $C_i \subseteq V$ is called a *community* and we wish to have more edges connecting vertices in the same communities than edges that connect vertices in different communities. The *modularity* of $\mathcal{C}$ is the fraction of the edges that fall within the given communities minus the expected number of such fraction if edges were distributed at random. The randomization of the edges is done so as to preserve the degree of each vertex. If vertices i and j have degrees d_i and d_j, then the expected number of edges between i and j is $\frac{d_i d_j}{2M}$. Thus, the modularity, denoted by Q, is then

$$Q(\mathcal{C}) = \frac{1}{2M} \sum_{ij} (A_{ij} - \frac{d_i d_j}{2M}) \delta_{ij} \tag{14}$$

where M is the total edge weights and the element δ_{ij} of the membership matrix $\boldsymbol{\delta}$ is defined as

$$\delta_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are in the same community} \\ 0, & \text{otherwise} \end{cases}$$

The modularity values can be either positive or negative and the higher (positive) modularity values indicate stronger community structures. Therefore, the *maximizing modularity problem* asks us to find a division $\mathcal{C}$ which maximizes the modularity value $Q(\mathcal{C})$.

This problem is different from the partition problem as we do not know the total number of partitions beforehand. That being said, l is unspecified. Somewhat surprisingly, modularity maximization is still NP-complete on trees, one of the simplest graph classes.

Theorem 5 *Modularity maximization on trees is NP-complete.*

The proof has been presented in [23], reducing from the Subset-Sum problem.

Exact Solutions. Although the problem is in NP class, efficient algorithms to obtain optimal solutions for small size networks are still of interest. We have presented an exact algorithm with a run time of $O(n^5)$ to the problem on uniform-weighted trees [23]. The algorithm is based on the dynamic programming, which exploits the relationship between maximizing modularity and minimizing the sum-of-squares of component volumes, where volume of a component S is defined as $\mathsf{vol}(S) = \sum_{v \in S} d_v$.

When the input graph is not a tree, we provided an exact solution based on Integer Linear Programming (ILP) [23]. Note that in the ILP for modularity maximization, there is a triangle inequality $x_{ij} + xjk - x_{ik} \geq 0$ to guarantee the values of x_{ij} be consistent to each other. Here $x_{ij} = 0$ if i and j are in the same community; otherwise $x_{ij} = 0$. Therefore, the ILP has $3\binom{n}{3} = \theta(n^3)$ constraints, which is about half a million constraints for a network of 100 vertices. As a consequence, the sizes of solved instances were limited to few hundred nodes. Along this direction, we have presented a sparse metric, which reduces the number of constraints to $O(n^2)$ in sparse networks where $m = O(n)$.

Approximation Algorithms. When G is a tree, the problem can be solved by a polynomial time approximation scheme (PTAS) with a run time of $O(n^{\epsilon+1})$ for $\epsilon > 0$ [23]. The PTAS is solely based on the following observation. Removing $k-1$ edges in G will yields k connected communities and $Q_k \geq (1 - \frac{1}{k})Q_{opt}$ where Q_k is the maximum modularity of a community structure with k communities, and Q_{opt} is the optimal solution.

When G having the degree distribution follows the power-law, i.e., the fraction of nodes in the network having k degrees is proportional to $k^{-\gamma}$, where $1 < \gamma \leq 4$, the problem can be approximated to a constant factor for $\gamma > 2$ and up to an $O(1/\log n)$ when $1 < \gamma \leq 2$ [19]. The details of this algorithm, namely Low-Degree Following (LDF), is presented below.

Algorithm. Low-degree Following Algorithm (Parameter $d_0 \in \mathbb{N}^+$) 12pt

1. $L := \emptyset, M := \emptyset, O := \emptyset, p_i = 0 \; \forall i = 1..n$
2. **for each** vertex $i \in V$ **do**
3. **if** $(k_i \leq d_0)$ & $(i \notin L \cup M)$ **then**
4. **if** $N(i) \setminus M \neq \emptyset$ **then**
5. Select a vertex $j \in N(i) \setminus M$
6. Let $M = M \cup \{i\}, L = L \cup \{j\}, p_i = j$
7. **else**
8. Select a vertex $t \in N(i)$
9. $O = O \cup \{i\}, p_i = t$
10. $\mathcal{L} = \emptyset$
11. **for each** vertex $i \in V \setminus (M \cup O)$ **do**
12. $C_i = \{i\} \cup \{j \in M \mid p_j = i\} \cup \{t \in O \mid p_{p_t} = i\}$
13. $\mathcal{L} = \mathcal{L} \cup \{C_i\}$
14. Return $\mathcal{L}$

The selection of d_0 is important to derive the approximation factor as d_0 needs to be a sufficient large constant that is still relative small to n when n tends to infinity. In an actual implementation of the algorithm, we have designed an automatic selection of d_0 to maximize Q. LDF can be extended to solve the problem in directed graphs [19].

Furthermore, in some cases, communities are sharing some nodes between them, referred as overlapping communities. That is, a person or a node can belong to more than one community. Therefore, we further designed an algorithm to find overlapping network modules which required only one parameter, indicating the level of overlapping. Simulations showed that this is the best one in the literature. This work is published in the IEEE Conference on Social Computing, 2011.

2.2.2 Detailed Results for Adaptively Updating Community Structures

We continued studying the adaptive identification of community structures, focused on the following question: How to update the evolving community structures without re-computing it. In this approach, the community structure (CS) at time t is detected based on the community structure at time $t-1$ and the changes in the network, instead of recomputing it directly at time t without taking advantages of a current solution at time $t-1$. Along this direction, we have devised an adaptive approximation algorithm for this problem, published in [11]. Indeed, the above LDF algorithm can be enhanced to cope with this situation. At first LDF is run to find the base CS at time 0. Then at each time step, we adaptively follow and unfollow the nodes that violate the condition 3 in Alg LDF.

We further investigated the overlapping community structure and provided the first adaptive algorithm to adaptively updating the overlapping network modules. This work is published in [8, 56].

2.2.3 Detailed Results for Assessing Network Structure Vulnerability

Impact of Nodes' Failures on Network Components. In this task, we are interested in identifying the set of nodes whose removal triggers a significant restruction of the current community structure. In term of notations, given the input network, the community detection algorithm $\mathcal{A}$ and a positive number k, we formulated the *Community structure Vulnerability Assessment* (CVA) which aims to find a set S of k nodes whose removal maximally transforms the current network community structure to a totally different one, evaluated via the Normalized Mutual Information measure.

Definition 9 *Given a network represented by an undirected and unweighted graph G, a specific community detection algorithm $\mathcal{A}$, and a positive integer $k \leq N$, we seek for a subset $S \subseteq V$ such that $S = \underset{S' \subseteq V, |S'|=k}{\operatorname{argmin}} \{NMI_X(S')\}$, where $X \equiv \mathcal{A}(G)$, and $NMI_X(S') \equiv NMI(X, \mathcal{A}(G[V \backslash S']))$ for any $S' \subseteq V$.*

Our major findings of this tasks are: (1) We analyzed conditions that can possibly lead to the minimization of NMI on community structures. (2) We devised an approximation algorithm for the case $k = 1$, and suggested multiple heuristic algorithms for CVA problem. We validated the effectiveness of our solutions on both synthesized data with known community structures and real-world traces including Arxiv citation network, Facebook, and Foursquare social networks. The details can be found in [6].

We have provided the basic results for the MNI analysis as follows:

Lemma 7 *There is a graph $G = (V, E)$ in which $NMI_X()$ is not a submodular function. Moreover, there are subsets $L \subseteq T \subseteq V$ such that $NMI_X(T) \geq NMI_X(L)$ (where L, T are sets of removed nodes).*

Theorem 6 *Given two community assignments $A \subseteq B$, there is $s \notin A, B$ such that $NMI_X(A + x) - NMI_X(A) < NMI_X(B + s) - NMI_X(B)$.*

We provided three algorithms to find a subset S. Our first heuristic algorithm is oriented based on the modularity contributions of network communities in G. There are two versions in general, called $greedyM_N$ and $greedyM_C$, for this heuristic approach with different priorities given to nodes and communities. In $greedyM_N$, all nodes u's in the network are ranked based on their modularity contributions $q_{u,C}$'s, and the top k nodes are selected in the solution set. The second algorithms, $greedyM_C$, consists of two steps: it first finds the community C having the most modularity contribution q_C, and then selects a node u that has the highest modularity portion $q_{u,C}$ in C until k nodes are included in the solution set.

Our second heuristic approach $greedyC$ for CVA problem is based on the component. Basically, given a community structure X and the algorithm $\mathcal{A}$, $greedyC$ tries to find nodes which can potentially break current communities into smaller ones of the relatively same size, where the preference given to large-size communities. In particular, $greedyC$ looks into communities X_i's of X, ordered by their sizes, and selects nodes that can divide this community into more subcomponents.

Impact of Edges' Failures on Network Components. In this task, we are interested in identifying the set of edges whose removal triggers a significant reconstruction of the current community structure, defined as follows:

Definition 10 *(DBC) Given an undirected graph $G = (V, E)$, and a set $\mathscr{C}$ of k communities, find a subset $S \subset E$ of minimum cardinality such that removing S from the graph breaks every community in $\mathscr{C}$.*

Our major findings of this task are:

- We defined the framework for community structure fragility. At first we introduced the density based broken community (DBC) problem for breaking k communities with the minimum number of edge removals and provided an approximation algorithm, namely CVA, with theoretical performance guarantee, $O(\log k)$. Its pseudo-cde is shown in Algorithm 8.
- To analyze the vulnerability of the community structures in a broader sense, we extended the problem formulation to communities produced from an *arbitrary* community detection algorithm. We offered an efficient heuristic to break the communities and identify the set of critical edges.
- We conducted extensive experiments with different parameters to mine interesting observations about the behavior of broken communities after edge removal.

The details can be found in [2].

For general definition of community structure, we extended the DBC problem to the following one:

Definition 11 *(Broken Community) Consider a community detection algorithm $\mathscr{A}$, which produces a collection $\mathscr{C}$ of communities on graph G (written $\mathscr{C} = \mathscr{A}(G)$). Let G' be a new graph after removal of a set of edges, and let $\mathscr{C}' = \mathscr{A}(G')$. Let $\gamma \in (0, 1)$. A community $C \in \mathscr{C}$ is said to be* broken *in graph G' if there does not exist a community $C' \in \mathscr{C}'$ satisfying*
*(*i*) $C' \subset C$, and (*ii*) $|C'|/|C| > \gamma$*

We have shown that partitioning a community C into at least c ϵ-balanced subparts, where $\gamma c \geq 1 + \epsilon$ makes it broken. Therefore, we developed the following Algorithm 9.

2.3 Impact Analysis of the Power-Law Degree Distribution on Network Vulnerability

Many practical complex networks, such as the Internet and WWW are discovered to follow power-law distribution in their degree sequences, i.e., the number of nodes with degree i in these networks is proportional to $i^{-\beta}$ for some exponential factor $\beta > 0$. Although power-law networks have been found robust under random attacks and vulnerable to intentional attacks via experimental observations, a better understand of their vulnerabilities from a theoretical point of view still remains open.

Furthermore, it is a common belief that solving solve optimization problems in power-law graphs is easier that than in general graphs. From an algorithmic perceptive, some experiments

Algorithm 8: CVA: An approximation algorithm for finding the critical edges

Data: Network $G = (V, E)$, $DeletionVector\ D$, $\mathscr{C}$, $|\mathscr{C}| = k$
Result: A set $S \subseteq E$ edges

$S \leftarrow \emptyset$;
$C \leftarrow \emptyset$;
for *each edge* $e \in E$ **do**
    compute the gain $f(e)$;
end
while $|C| < k$ **do**
    $e' \leftarrow \underset{e \in E}{\operatorname{argmax}}\{f(e)\}$;
    In case of a tie, choose randomly;
    $S \leftarrow S \cup \{e'\}$;
    for $l = 1$ *to* k **do**
        if $C_l \notin C$ **then**
            if $e' \in C_l$ **then**
                $D_l \leftarrow D_l - 1$;
                if $D_l \leq 0$ **then**
                    $C \leftarrow C \cup \{C_l\}$;
                    $f(e) = f(e) - 1$ for all $e \in C_l$;
                end
            end
        end
    end
end
return S;

Algorithm 9: CCF: A heuristic algorithm for breaking communities

Data: Network $G = (V, E)$, k Communities $\mathscr{C}$, strictness threshold γ
Result: A set $S \subseteq E$ of edges

$S \leftarrow \phi$;
$c \leftarrow z : z$ is least integer satisfying $z\gamma \geq 1 + \epsilon$;
for *each community* $C_i \in \mathscr{C}$ **do**
    compute the c-way balanced partitioning;
    $Cut_i =$ set of edges to cut C_i into c parts;
    $S \leftarrow S \cup Cut_i$;
end
return S;

had been developed to evaluate the simple algorithms for optimization algorithm in power-law graphs. However, there is no work that provides an algorithm framework for solving a set of problems in power-law graphs with the degree distribution property, let alone a theoretical analysis framework for analyzing approximation ratios.

Therefore, we focused on addressing the following issues: (i) Analyzed the power-law network vulnerability under various attacks, (ii) Studied the complexity of many optimization problems, and (iii) Developed approximation algorithms on power-law graphs.

Relevance. Clearly this study is very important in understanding of and providing solutions to network vulnerability in real-life as many of them follows the power-law degree distribution. The results in this task advance the research front of approximation theory and optimization. They help us develop several solutions for many problems on PLNs, and thus the findings in this task are extremely helpful for many applications.

Summary of Findings.

- Showed that the power-law networks almost surely are not vulnerable to attacks, including random and preferential attacks when β is small enough.
- Developed a new embedding technique to re investigate most of classic optimization problem on power-law networks such as dominating set, maximum independent set, vertex cover, clique. We have shown these problems remain NP-complete on power-law networks but they may have better (tighter) approximation ratios.
- Developed an approximation algorithm framework, called Low-Degree Percolation (LDP) Algorithm Framework, for solving Minimum Dominating Set (MDS), Minimum Vertex Cover (MVC) and Maximum Independent Set (MIS) problems in power-law networks.

2.3.1 Detailed Results for Vulnerability Analysis

In this task, we studied the vulnerability of power-law networks under random attacks and adversarial attacks using the in-depth probabilistic analysis on the theory of random power-law graph models. Our results indicate that power-law networks are able to tolerate random failures if their exponential factor β is less than 2.9, and they are more robust against intentional attacks if β is smaller. Furthermore, we revealed the best range $[1.8, 2.5]$ for the exponential factor β by optimizing the complex networks in terms of both their vulnerabilities and costs. When $\beta < 1.8$, the network maintenance cost is very expensive, and when $\beta > 2.5$ the network robustness is unpredictable since it depends on the specific attacking strategy.

The detailed proofs can be found in [37, 38]. Here, we listed our major findings in terms of the theorems.

Theorem 7 *In a residual graph G_r of $G_{(\alpha,\beta)}$ after random failures,*

- *If $\beta < \beta_p$, the expected pairwise connectivity $E(\mathbb{P})$ is a.s. $\Theta(n^2)$;*
- *If $\beta \geq \beta_p$, the pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{2}n\left(c_r(\beta)n^{\frac{2}{\beta}}\log n - 1\right)$.*

where β_p satisfies that $(1-p)\zeta(\beta_p-2)-(2-p)\zeta(\beta_p-1)=0$ and

$$c_r(\beta)=16/\left[\zeta(\beta)\left(2-p-(1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)}\right)\right]^2$$

.

Theorem 8 *In a residual graph G_p^I of $G_{(\alpha,\beta)}$ after interactive preferential attacks,*

- *If $\beta+\beta'<3.47875$, the expected pairwise connectivity $E(\mathbb{P})$ is $\Theta(n^2)$;*
- *If $\beta+\beta'\geq 3.47875$, the pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{2}n\left(c(\beta)n^{\frac{2}{\beta}}\log n-1\right)$.*

where $c(\beta)=16/\left[\zeta(\beta)\left(2-\frac{\zeta(\beta-2)}{\zeta(\beta-1)}\right)\right]^2$ is a constant on any given β.

Theorem 9 *In a residual graph G_p^E of $G_{(\alpha,\beta)}$ after expected preferential attacks,*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $c<\min\left\{c\left|\frac{\sum_x \frac{e^\alpha}{x^\beta}\left(1-\frac{cx}{e^\alpha\zeta(\beta-1)}\right)\left(x\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)\right)}{n-c}>1\right.\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4}n^{\frac{3}{2}}\log n$ if $c>\max\left\{c\left|\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)\frac{\zeta(\beta-2)-\frac{c\zeta(\beta-3)}{e^\alpha\zeta(\beta-1)}}{\zeta(\beta-1)-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)}}<1\right.\right\}$.*

Theorem 10 *In a residual graph G_c of $G_{(\alpha,\beta)}$ after degree-centrality attacks,*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $x_0>\min\left\{x_0\left|\frac{1}{\zeta(\beta-1)}\frac{\left(\sum_{x=1}^{x_0}\frac{1}{x^{\beta-1}}\right)^2}{\sum_{x=1}^{x_0}\frac{1}{x^\beta}}>1\right.\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4}n^{\frac{3}{2}}\log n$ if $x_0<\max\left\{x_0\left|\frac{1}{\zeta(\beta-1)}\sum_{x=1}^{x_0}\frac{1}{x^{\beta-2}}<1\right.\right\}$.*

Lemma 8 *Let G_{cp} be the residual graph of $G_{(\alpha,\beta)}$ only consisting of the protected degree-centrality nodes (the nodes of degree larger than x_0), we have*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $x_0<\max\left\{x_0\left|\frac{1}{\zeta(\beta-1)}\frac{\left(\sum_{x=x_0+1}^{\Delta}\frac{1}{x^{\beta-1}}\right)^2}{\sum_{x=x_0+1}^{\Delta}\frac{1}{x^\beta}}>1\right.\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4}n^{\frac{3}{2}}\log n$ if $x_0>\min\left\{x_0\left|\frac{1}{\zeta(\beta-1)}\sum_{x=x_0+1}^{\Delta}\frac{1}{x^{\beta-2}}<1\right.\right\}$.*

Theorem 11 *In the residual graph G_s of $G_{(\alpha,\beta)}$, the expected size of a connected component $\bar{c}$ is a.s. upper bounded by $O\left(n^{\frac{1}{4}}\right)$ when $\tilde{d}_s<1$, that is, $x_0<\max\left\{x_0\left|\frac{1}{\zeta(\beta-1)}\sum_{x=1}^{x_0}\frac{1}{x^{\beta-2}}<1\right.\right\}$.*

2.3.2 Detailed Results for Hardness Complexity

In this task, we aimed to develop new embedding techniques for many classical problems to investigate whether the are remained in NP-complete even on power-law graphs.

Embedding techniques allow an original graph G to embed to a power-law graph G_p such that a considered problem can be polynomially solved in $G_p \setminus G$, thus preserving the complexity of the problem on G to G_p. We developed two new techniques on optimal substructure problems, *Cycle-Based Embedding Technique* and *Graphic Embedding Technique*, to embed a d-bounded graph into a general power-law graph and a simple power-law graph respectively. Then we used these two techniques to further proved the *APX*-hardness and the inapproximability of many classical problems such as Minimum Dominating Set (MDS), Maximum Independent Set (MIS), Minimum Dominating Set (MDS), Minimum Vertex Cover (MVC), Clique, Coloring, ρ-Minimum Dominating Set (ρ-MDS) on general power-law graphs and simple power-law graphs. These inapproximability results on power-law graphs are shown in Table 1.

Table 1: Inapproximability Factors on Power-Law Graphs with Exponential Factor $\beta > 1$

Problem	General Power-Law Graph	Simple Power-Law Graph
MIS	$1 + \frac{1}{140(2\zeta(\beta)3^{\beta}-1)} - \varepsilon$	$1 + \frac{1}{1120\zeta(\beta)3^{\beta}} - \varepsilon$
MDS	$1 + \frac{1}{390(2\zeta(\beta)3^{\beta}-1)}$	$1 + \frac{1}{3120\zeta(\beta)3^{\beta}}$
MVC, ρ-MDS	$1 + \frac{2\left(1-(2+o_c(1))\frac{\log\log c}{\log c}\right)}{\left(\zeta(\beta)c^{\beta}+c^{\frac{1}{\beta}}\right)(c+1)}$	$1 + \frac{2-(2+o_c(1))\frac{\log\log c}{\log c}}{2\zeta(\beta)c^{\beta}(c+1)}$
CLIQUE	-	$O\left(n^{1/(\beta+1)-\epsilon}\right)$
COLORING	-	$O\left(n^{1/(\beta+1)-\epsilon}\right)$

[a] Conditions: MIS and MDS: P≠NP; MVC, ρ-MDS: unique games conjecture; CLIQUE, COLORING: NP≠ZPP.

The inapproximablity results show that it is easier to find the solution for these problems on PLNs than that of on the general graphs. The details of cycle-based embedding techniques and graphic embedding technique can be found in [43, 45]. Here, we list the major theorems:

Theorem 12 (Cycle-Based Embedding Technique) *Any d-bounded graph G_d can be embedded into a power-law graph $G_{(\alpha,\beta)}$ with $\beta > 1$ such that G_d is a maximal component and most optimal substructure problems can be polynomially solvable on $G_{(\alpha,\beta)} \setminus G_d$.*

Lemma 9 *Given a sequence of integers $D = \langle d_1, d_2, \ldots, d_n \rangle$ which is non-increasing, continuous and the number of elements is at least as twice as the largest element in D, i.e. $n \geq 2d_1$, it is possible to construct a simple graph G whose d-degree sequence is D in polynomial time $O(n^2 \log n)$.*

Theorem 13 (Graphic Embedding Technique) *Any d-bounded graph G_d can be embedded into a simple power-law graph $G_{(\alpha,\beta)}$ with $\beta > 1$ in polynomial time such that G_d is a maximal component and the number of vertices in $G_{(\alpha,\beta)}$ can be polynomially bounded by the number of vertices in G_d.*

2.3.3 Detailed Results for Approximation Algorithms

Due to the above results, we continued to develop a general approximation framework, called Low-Degree Percolation (LDP), to approximate the optimization problems in power-law networks, including MIS, MDS, and MVC problems. The idea of LDP framework is to percolate the graph starting from a large number of low-degree nodes in a power-law graph, which allows us to develop a theoretical framework, which can be used to analysis the approximation ratios via probabilistic analysis.

In particular, we applied this theoretical framework to show the approximation ratios for these problems on two well-known random power-law models as follows:

Theorem 14 (Main Theorem (MDS&MVC)) *In a power-law graph $G_{(\alpha,\beta)}$, by using LDP Algorithm, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least $1 - p_\lambda$, where Ψ is the approximation ratio of MDS (or MVC) in general graphs w.r.t. a graph of size at most $e^\alpha \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$.

Theorem 15 (Main Theorem (MIS)) *In a power-law graph $G_{(\alpha,\beta)}$, by using LDP Algorithm, MIS can be approximated into*

$$\frac{N + e^\alpha\left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}{N + \frac{1}{\Psi} e^\alpha\left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}$$

with probability at least $1 - p_\lambda$, where N is the number of nodes with degree 1, Ψ is the approximation ratio of MIS in general graphs w.r.t. a graph of size at most $e^\alpha \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$.

Theorem 16 *In a Structural Random Power-Law (SRPL) Graph G, by using LDP Algorithm, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least

$$\sum_{\tau=0}^{\lfloor e^\alpha/2 \rfloor} Pr[C_2 = \tau](1 - p_\lambda^\tau)$$

where $p_\lambda^\tau = \frac{1}{\frac{(\lambda^\tau - \mu(\alpha,\beta,2))^2}{\chi(\alpha,\beta,2)} + 1}$ in which $\lambda^\tau = \lambda + \frac{\tau}{\sum_{i=2}^{\Delta} \frac{1}{i^\beta}}$.

The major findings of this task include: (i) Developed an approximation algorithm framework, called Low-Degree Percolation (LDP) Algorithm Framework, for solving Minimum Dominating Set (MDS), Minimum Vertex Cover (MVC) and Maximum Independent Set (MIS) problems in power-law graphs. (ii) Using this framework, we further showed a theoretical framework to derive the approximation ratios for these optimization problems in two well-known random power-law graphs. (iii) Our numerical analysis showed that, these optimization problems can be approximated into near 1 factor with high probability, using our proposed LDP algorithms, in power-law graphs with exponential factor $\beta \geq 1.5$, which belongs to the range of most real-world networks. The details of these findings can be found in [17].

In details, the approximation algorithm framework to solve optimization problems is based on the degree sequence property in power-law graphs. As can be seen, the most fundamental property of power-law graphs is that they contain a large number of low-degree nodes, while only a small number of high-degree nodes. Therefore, the idea of our proposed Low-Degree Percolation (LDP) algorithm framework is to sort the nodes by their degrees and percolate the graph from the nodes of lowest degree. The process continues in residual graph iteratively until no more nodes, which are surely in optimal solution, can be detected. At last, we applied existing approximation approaches in the remaining graph.

For MDS and MVC problems, as shown in Algorithm 10, since the node incident to a node of degree 1 certainly belongs to an optimal solution, we percolated the graph by adding all the neighbors of nodes with degree 1 in each iteration. Until no more nodes of degree 1 exists in residual graph, we applied existing approximation algorithm to obtain the solution in this residual graph.

Algorithm 10: LDP Algorithm for MDS/MVC Problems

Input : Power-law graph G
Output: MDS (or MVC) S
while $\exists$ *Nodes of degree 1* **do**
  foreach *Node* v *of degree 1* **do**
    Add its neighbor $N(v)$ into S;
    Remove v from G;
  end
  Remove all nodes incident to S from graph G;
end
Determine the leftover MDS (or MVC) in G using existing approximation algorithm and add them into S;
return S;

On the other hand, Algorithm 11 shows the algorithm for MIS. In this case, the nodes of degree 1 will belong to the optimal solution, and in the meanwhile, it is certain that their neighbors cannot be in optimal solution any more. Therefore, in order to obtain MIS, we selected all nodes of degree 1 into the solution in each iteration and ran an existing approximation algorithm to obtain the MIS in the remaining graph.

We have proved its approximation ratio in the following Theorems:

Theorem 17 (Main Theorem (MDS&MVC)) *In a power-law graph $G_{(\alpha,\beta)}$, by using Algorithm 10, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least $1 - p_\lambda$, where Ψ is the approximation ratio of MDS (or MVC) in general graphs w.r.t. a graph of size at most $e^\alpha \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$.

And for the structural random power-law (SRPL) graph, we obtained the following results:

Algorithm 11: LDP Algorithm for MIS Problem

Input : Power-law graph G
Output: MIS S

while $\exists$ *Nodes of degree 1* **do**
  foreach *Node* v *of degree 1* **do**
    Add v into S;
    Remove v and all its neighbors $N(v)$ from G;
  end
end
Determine the leftover MIS in G using existing approximation algorithm and add them into S;
return S;

Theorem 18 *In a SRPL graph G, by using Algorithm 10, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least

$$\sum_{\tau=0}^{\lfloor e^{\alpha}/2 \rfloor} Pr[C_2 = \tau](1 - p_{\lambda}^{\tau})$$

where $p_{\lambda}^{\tau} = \frac{1}{\frac{(\lambda^{\tau} - \mu(\alpha,\beta,2))^2}{\chi(\alpha,\beta,2)} + 1}$ *in which* $\lambda^{\tau} = \lambda + \frac{\tau}{\sum_{i=2}^{\Delta} \frac{1}{i^{\beta}}}$.

3 Training and Professional Development

3.1 Personnel Supported

This grant has partially supported the following personnel.

- PI: My T. Thai
- Ying Xuan (Ph.D, graduated in Fall 2011, currently is a Research Staff at IBM
- Thang N. Dinh (Ph.D, graduated in Spring 2013, currently is an Assistant Professor at Virginia Commonwealth University)
- Yilin Shen (Ph.D, graduated in Spring 2013, currently is a Research Scientist at Samsung America, Research Center)
- Nam Nguyen (Ph.D, graduated in Spring 2013, currently is an Assistant Professor at Towson University)
- Dung Nguyen (Ph.D, graduated in Summer 2013, currently is at Microsoft)
- MS Students: Yu-Song Syu (graduated in Spring 2012)
- Subhankar Mishra (current Ph.D. student, working towards his degree.)

3.2 Training

Several of my Ph.D students have been working with me on this project, thus it of course provides them several advanced professional skills in doing research and their own professional developments as shown above. They have been trained to a new set of problems along with the developing of several fundamental theories and algorithms. The results of these works have been included in these PhD students' dissertations (please see Appendix). Many of these students have obtained either faculty positions or research staffs in the industry.

The students have attended several conferences to present their papers, such as IEEE INFOCOM 2010, IEEE MILCOM 2011, ACM Mobicom 2011, COCOON 2011, IEEE SocialCom 2011, IEEE INFOCOM 2011, ACM CIKM 2012, ACM HyperText 2012, ACM WebSci 2012, ACM ICDM 2013, ACM ASONAM 2013, IEEE ICDCS 2013, IEEE GLOBECOM 2013, ACM WI 2014, and COCOON 2014.

Since the work is on the complex network, I have used several results here to offer a new course, namely Optimization in Adaptive Complex Systems and Social Networks. This course is a graduate level course, focusing on many problems arising in complex networks and systems, such that vulnerability, cascading failures, complex network models.

3.3 Professional Development

During the period of this project, the PI is an associate editor for Journal of Combinatorial Optimization, IEEE Transactions on Parallel and Distributed Systems, Journal of Discrete Mathematics, and optimization brief series editor for Springer. She is also a PC chair for several conferences, including IEEE ISSPIT 2012, IEEE IWCMC 2012, SIMPLEX 2011.

DySON 14, CSoNets 13, IEEE IWCMC 12, IEEE ISSPIT 12, SIMPLEX 11, DIS 11, COCOON 10, CCNet 10

The PI is a co-founder and EiC of a new Springer journal, namely Computational Social Networks. She has created and chair a workshop on mathematics of social networks ($http://www.cise.ufl.edu/\tilde{m}ythai/CSoNet.html$), and has been a PC members of many conferences, such as INFOCOM, SOCIALCOM, ICDCS. She is founding another workshop on interdependent networks, co-located with the first rank conference IEEE Infocom 2015, namely WIDN (pronounced as Widen). Information can be founded at $http://optnetsci.cise.ufl.edu/widn2015/$

The PI has been given several invited talks and seminars, listed as follows:

- "Interdependent Networks Analysis,", Learning and Intelligent Optimization Conference (LION 8), Feb 16-21, 2014, Florida. Tutorial Talk
- "Cybersecurity in an Era of Online Social Networks" Conference on Selected Problems on IT and Telecommunication, Nov 14-15, 2013, Danang, Vietnam. Plenary Keynote
- "Dynamic Community Structure Analysis in Complex Networks," the Int Conference on the Dynamics of Information Systems, February 25-27, 2013. USA. Plenary Keynote
- "Community Structure Analysis in Dynamic Complex Networks" the 9th AIMS Conference on Dynamic Systems, Differential Equations and Applications, July 1 - 5, 2012, USA. Invited Speaker

- "How the Power-law Distribution Impacts on the Complex Network Vulnerability" Int Workshop on Complex Networks, March 7-9, 2012, USA. Invited Speaker
- "Optimally Use of Social Networks to Manipulate Information", University of Texas Dallas, Department of Computer Science, September 2011. Department Colloquium talk

4 Results Dissemination

We have disseminated the results in several avenues:

- Published the results in IEEE/ACM proceedings and journals
- Participated and presented our findings at conference meetings and seminars
- Provided source codes of our algorithms per requested from several research groups
- Developed two important interactive web-based tools which tremendously help the researchers in the networking field to conduct their research based on community structure concepts.

  1. Identify community structure. Complex networks exhibit a community structure property, which is nodes within communities are densely connected than that between communities. Identifying community structures is a central topic of network science. This research is of significant importance as it provides insights into the functionality of a network and finds itself extremely useful in deriving social-based solutions. My group has developed a web-based tool which allows researchers to input any network of interests, and the tool will return a community structure of this input. The tool also allows users to interactively rearrange these communities for a better view.

  2. Break community structure with the minimum cost. Researchers believed that communities are very strong and it is hard to be broken as they are densely connected. However, I have shown that it is otherwise. In addition to the published theoretical results, I have developed an online interactive tool for researchers to see how and where to break the communities. This tool helps researchers gain an insight to the structure of studied networks in order to devise better solutions to protect such networks.

5 Honors/Awards

- The PI is a recipient of NSF CAREER award 2010-2015.
- The PI received the Provost's Excellence Award for Assistant Professors at the University of Florida, 2010
- The PI has been early promoted to the rank of Associate Professor in 2010
- Ph.D students Yilin Shen and Thang Dinh have received UFIC Outstanding International Student Awards

6 Dataset

We have evaluated our algorithms with various datasets, described in the following.

6.1 Critical Node Detections

The three networks which we use to evaluate the performance of our algorithms related to the Critical Node Detection problem and its variants are described as follows:

- The real terrorist network compiled by Krebs with 62 nodes and 153 links, which reflects the relationship between the terrorists involved in the terrorism attacks of Sep. 11, 2001. This experiment attempts to evaluate the performance of HILPR on a real-world social network. In order to breakdown the terrorist network, we can capture the individuals corresponding to the critical nodes identified by HILPR.
- Waxman network topology, a widely-accepted Internet AS topological model, is generated by the well-known BRITE.
- Power-law network topology, generated by Barabási graph generator
- Western States power network of the US with 4941 nodes and 6594 edges
- Small-world network topology generated by igraph library using Watts and Strogatz model, with $k = 2$, $\mu = 0.2$ and 70 nodes.
- US Network Assets compiled with 71 nodes and 98 edges, which provides the current customer needs in XO Communications service.
- *Erdos-Reyni*: A random graph of 100 vertices and 200 edges following the Erdos-Reyni model.
- *Forest fire*: A random power-law graph following Forest fire model by Leskovec et al. with the forward and backward burning probabilities 0.3 and 0.9, respectively.
- *US Backbone network*: The backbone cabling network of XO company.
- *CAIDA AS*: The CAIDA AS Relationships Dataset from Sep. 17, 2007.
- *Oregon AS*: AS peering information inferred from Oregon route-views between March 31 and May 26, 2001. Only the largest connected component with 11,174 nodes and 23,410 links is considered.
- *Gnutella P2P*: Gnutella peer-to-peer network from from Aug. 25, 2002. Nodes represents hosts in the network and edges are the connections between the Gnutella hosts. It consists of 22,663 nodes and 108,386 edges
- Coauthor network in Physics sections of the e-print arXiv
- Partial Facebook with 63K nodes and 817K edges
- Orkut with 3M nodes and 223M edges

6.2 Network Structural Interdependency and Vulnerability Assessment

To detect and verify the community structures, we evaluated our algorithms in the following dataset, described in the table.

Table 2: Order and size of network instances

ID	Name	Vertices (n)	Edges (m)
1	Zachary's karate club	34	78
2	Dolphin's social network	62	159
3	Les Miserables	77	254
4	Books about US politics	105	441
5	American College Football	115	613
6	US Airport 97	332	2126
7	Electronic Circuit (s838)	512	819
8	Scientific Collaboration	1589	2742

We also evaluated them on the dataset of social networks as mentioned in the above section, including Facebook, Twitter, Orkut, ENRON Email, ArXiv Citation,

7 Publications

1. Y. Shen and M. T. Thai, Vulnerability Assessment under Cascading Failures, in Proceedings of the IEEE Global Communication Conference (GLOBECOM), 2013

2. Md A. Alim, A. Kuhnle, and M. T. Thai, Are Communities As Strong As We Think?, in Proceedings of IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining (ASONAM), 2014.

3. S. Mishra, X. Li, M. T. Thai, and J. Seo, Cascading Critical Nodes Detection with Load Redistribution in Complex Systems, in Proceedings of the 8th Annual International Conference on Combinatorial Optimization and Application (COCOA), 2014

4. S. Mishra, T. N. Dinh, M. T. Thai, and I. Shin, Optimal Inspection Points for Malicious Attack Detection in Smart Grids, in Proceedings of the 20th Int Computing and Combinatorics Conference (COCOON), 2014

5. D. T. Nguyen, H. Zhang, S. Das, M. T. Thai, and T. N. Dinh, Least Cost Influence in Multiplex Social Networks: Model Representation and Analysis, in Proceedings of the IEEE Int Conference on Data Mining (ICDM), 2013.

6. Md. A. Alim, N. P. Nguyen, T. N. Dinh, and M. T. Thai, Vulnerability Analysis of Overlapping Communities in Complex Networks, in Proceedings of the 2014 IEEE/WIC/ACM International Conference on Web Intelligence (WI), 2014.

7. T. N. Dinh and M. T. Thai, Network under Joint Node and Link Attacks: Vulnerability Assessment Methods and Analysis, IEEE Transactions on Networking (ToN), DOI: 10.1109/TNET.2014.2317486, 2014

8. N. P. Nguyen, T. N. Dinh, Y. Shen, and M. T. Thai, Dynamic Social Community Detection and its Applications PLoS ONE 9(4): e91431. doi:10.1371/journal.pone.0091431, 2014.

9. T. N. Dinh, M. T. Thai, and H. Nguyen, Bound and Exact Methods for Assessing Link Vulnerability in Complex Networks, Journal of Combinatorial Optimization (JOCO), vol. 28, no. 1, pp. 3-24, 2014

10. M. Hemmati, J.C. Smith, and M. T. Thai, A Cutting-plane Algorithm for Solving a Weighted Influence Interdiction Problem, Computational Optimization and Applications, vol. 57, no. 1, pp 71–104, 2014.

11. T. N. Dinh, N. P. Nguyen, M. A. Alim, and M. T. Thai, A Near-optimal Adaptive Algorithm for Maximizing Modularity in Dynamic Scale-free Networks, Journal of Combinatorial Optimization (JOCO), DOI:10.1007/s10878-013-9665-1, pp. 1-21, Oct 2013

12. H. Zhang, S. Mishra, M. T. Thai, Recent Advances in Information Diffusion and Influence Maximization in Complex Social Networks, Opportunistic Mobile Social Networks, (J. Wu and Y. Wang eds), CRC Press, Taylor & Francis Group, 2014

13. T. N. Dinh and M. T. Thai, Computing and Assessing All-pairs End-to-end Network Reliability, manuscript, submitted to INFOCOM 2015

14. N. P. Nguyen, M. A. Alim, T. N. Dinh and M. T. Thai,A method to detect communities with stability in social networks, Social Network Analysis and Mining, Vol. 4, Issue 1, DOI: 10.1007/s13278-014-0224-2, 2014.

15. N. P. Nguyen, Md. A. Alim, Y. Shen, and M. T. Thai, Assessing Network Vulnerability in a Community Structure Point of View, in Proceedings of IEEE/ACM Int Conf on Advances in Social Networks Analysis and Mining (ASONAM), 2013.

16. Y. Shen, D. T. Nguyen, Y. Xuan, and M. T. Thai, New Techniques for Approximating Optimal Substructure Problems in Power-Law Graphs, Theoretical Computer Science (TCS), vol. 447, 2012.

17. Y. Shen, X. Li, and M. T. Thai, Approximation Algorithms for Optimization Problems in Random Power-Law Graphs, COCOA, 2014

18. Y. Shen and M. T. Thai, Network Vulnerability Assessment under Cascading Failures, in Proceedings of the IEEE Global Communication Conference, (GLOBECOM), 2013

19. T. N. Dinh and M. T. Thai, Community Detection in Scale-free Networks: Approximation Algorithms for Maximizing Modularity, IEEE Journal on Selected Areas in Communications: Special Issue on Network Science (JSAC), vol. 31, no. 6, pp. 997–1006, June 2013.

20. M. T. Thai, R. Tiwari, R. Bose, and A. Helal, On Detection and Tracking of Variant Phenomena Clouds, IEEE Transactions on Sensor Networks (ToSN), vol. 10, no. 2, 2013.

21. N. P. Nguyen, G. Yan, and M. T. Thai, Analysis of Misinformation Containment in Online Social Networks, Elsevier Computer Networks-Towards a Science of Cyber Security (COMNETS), vol. 57, no. 10, pp. 2133–2146, July 2013

22. T. N. Dinh, Y. Shen, D. T. Nguyen, and M. T. Thai, Cost-effective Viral Marketing for Time-critical Campaigns in Large-scale Social Networks, IEEE/ ACM Transactions on Networking (ToN),), vol. pp, no. 99, 2013

23. T. N. Dinh and M. T. Thai, Towards Optimal Community Detection: From Trees to General Weighted Networks, Internet Mathematics, DOI:10.1080/15427951.2014.950875, 2014

24. H. Zhang, T. N. Dinh, and M. T. Thai, Maximizing the Spread of Positive Influence in Online Social Networks, in Proceedings of the IEEE Int Conference on Distributed Computing Systems (ICDCS), 2013.

25. Y. Shen, D. T. Nguyen, and M. T. Thai, Adaptive Approximation Algorithms for Hole Healing in Hybrid Wireless Sensor Networks, in Proceedings of the IEEE Int Conference on Computer Communications (INFOCOM), 2013

26. T. N. Dinh, N. P. Nguyen, and M. T. Thai, An Adaptive Approximation Algorithm for Community Detection in Dynamic Scale-free Networks, in Proceedings of the IEEE Int Conference on Computer Communications (INFOCOM) Mini-Conference, 2013.

27. T. N. Dinh, Y. Shen, and M. T. Thai, The Walls Have Ears: Optimize Sharing for Visibility and Privacy in Online Social Networks, in Proceedings of ACM Int Conference on Information and Knowledge Management (CIKM), 2012.

28. Y. Shen, T. N. Dinh, H. Zhang, and M. T. Thai, Interest-Matching Information Propagation in Multiple Online Social Networks, in Proceedings of ACM Int Conference on Information and Knowledge Management (CIKM), 2012

29. T. N. Dinh, Y. Shen, and M. T. Thai, An Efficient Spectral Bound for Link Vulnerability Assessment in Large-scale Networks, IEEE MILCOM, 2012

30. Y. Shen, T. N. Dinh, and M. T. Thai, Adaptive Algorithms for Detecting Critical Links and Nodes in Dynamic Networks, IEEE MILCOM, 2012

31. Y. Shen, T. Nguyen, and M. T. Thai, On the Discovery of Critical Nodes and Links for Assessing Network Vulnerability, IEEE Trans on Networking, , IEEE Transactions on Networking (ToN), vol. 21, no. 3, 2013

32. D. Nguyen, Y. Shen, and M. T. Thai, Detecting Critical Nodes in Interdependent Power Networks for Vulnerability Assessment, IEEE Trans on Smart Grid, vol. 4, no. 1, pp. 151–159, 2012

33. T. Dinh, Y. Shen, and M. T. Thai, On the Approximability of Positive Influence Dominating Set in Complex Networks, Journal of Combinatorial Optimization, vol. 27, no. 3, pp. 487–503, 2014

34. T. N. Dinh, D. T. Nguyen, and M. T. Thai, A Unified Approach for Domination Problems on Different Network Topologies, Handbook of Combinatorial Optimization, (P. Pardalos, D.-Z. Du, and R. Graham eds), Springer Publisher, 2012

35. T. N. Dinh, D. T. Nguyen, and M. T. Thai, Cheap, Easy, and Massively Effective Viral Marketing in Social Networks: Truth or Fiction?, ACM Conference on Hypertext and Social Media (Hypertext), 2012.

36. D. T. Nguyen, N. P. Nguyen, and M. T. Thai, Sources of Misinformation in Online Social Networks: Who to Suspect?, IEEE MILCOM, 2012

37. Y. Shen, N. P. Nguyen, and M. T. Thai, Exploiting the Robustness on Power-Law Networks, Int Computing and Combinatorics Conference (COCOON), 2011.

38. Y. Shen, N. Nguyen, and M. T. Thai, The Robustness of Power-Law Networks: Its Assessment and Optimization, IEEE Trans on Reliability, accepted with revision, 2012

39. Y. Xuan, Y. Shen, and N. P. Nguyen, Efficient Multi-Link Failure Localization in All-Optical Networks, IEEE Transactions on Communications (TCOM), vol. 61, no. 3, pp. 1144 – 1151, 2013

40. M. T. Thai and P. Pardalos (eds), Handbook of Optimization in Complex Networks: Theory and Applications, Springer Publisher, 2011, ISBN: 978-1461407539

41. M. T. Thai and P. Pardalos (eds), Handbook of Optimization in Complex Networks: Communication and Social Networks, Springer Publisher, 2011, ISBN: 978-1461408567

42. M. T. Thai, T. N. Dinh, and Y. Shen, Hardness and Approximation of Network Vulnerability, Handbook of Combinatorial Optimization, (P. Pardalos, D.-Z. Du, and R. Graham eds), Springer Publisher, ISBN 978-1-4419-7996-4, 2013

43. Y. Shen, D. T. Nguyen, and M. T. Thai, Hardness Complexity of Optimal Substructure Problems on Power-Law Graphs, Handbook of Optimization in Complex Networks: Theory and Applications, (M. T. Thai and P. Pardalos eds), Springer Publisher, 2011

44. N. P. Nguyen, Y. Xuan, and M. T. Thai, On Detection of Community Structure in Dynamic Social Networks, Handbook of Optimization in Complex Networks: Communication and Social Networks, (M. T. Thai and P. Pardalos eds), Springer Publisher, 2011

45. Y. Shen, D. T. Nguyen, and M. T. Thai, On the Hardness and Inapproximability of Optimization Problems on Power Law Graphs, in Proceedings of the Int Conference on Combinatorial Optimization and Applications (COCOA), 2010.

46. D. T. Nguyen, N. P. Nguyen, M. T. Thai, and S. Hela, Optimal and Distributed Algorithms for Coverage Hole Healing in Hybrid Sensor Networks, International Journal of Sensor Networks (IJSNet), vol. 11, no. 4, 2012

47. T. N. Dinh, Y. Xuan, M. T. Thai, P. Pardalos, and T. Znati, On New Approaches of Assessing Network Vulnerability: Hardness and Approximation, IEEE/ACM Transactions on Networking (ToN),), vol. 20, no. 2, pp. 609 – 619, 2012

48. Y. Xuan, Y. Shen, N. P. Nguyen, and M. T. Thai, A Trigger Identification Service for Defending Reactive Jammers in WSN, IEEE Transactions on Mobile Computing (TMC), vol. 11, no. 5, pp. 793–806, 2012

49. N. P. Nguyen and M. T. Thai, Finding Overlapped Communities in Online Social Networks with Nonnegative Matrix Factorization, in Proceedings of the IEEE Military Communications Conference (MILCOM), 2012

50. T. N. Dinh and M. T. Thai, Precise Structural Vulnerability Assessment via Mathematical Programming, in Proceedings of the IEEE Military Communications Conference (MILCOM), 2011.

51. T. Dinh, Y. Xuan, and M. T. Thai, Towards Social-Aware Routing in Dynamic Communication Networks, IPCCC 2010

52. T. Dinh, Y. Xuan, M. T. Thai, and T. Znati, On Approximation of New Optimization Methods for Assessing Network Vulnerability, INFOCOM 2010

53. M. T. Thai and T. Dinh, Hardness and Approximation of Network Vulnerability, Handbook of Combinatorial Optimization (D.-Z. Du, P. Pardalos, and R. Graham eds), Springer Publisher, ISBN 978-1-4419-7996-4, 2013

54. N. Nguyen, Y. Xuan, and M. T. Thai, A Novel Method on Worm Containment on Dynamic Social Networks, MILCOM 2010

55. Y. Xuan, Y. Shen, and M. T. Thai, A Graph-theoretic QoS Vulnerability Assessment for Network Topologies, GLOBECOM 2010

56. N. P. Nguyen, T. N. Dinh, S. Tokala, and M. T. Thai, Overlapping Communities in Dynamic Networks: Their Detection and Mobile Applications, in Proceedings of ACM International Conference on Mobile Computing and Networking (MobiCom), 2011.

Appendix

COMMUNITY STRUCTURE AND ITS APPLICATIONS IN DYNAMIC COMPLEX NETWORKS

By

NAM P. NGUYEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

ACKNOWLEDGMENTS

I would like express the deepest appreciation to my committee chair, Professor My T. Thai for always being a great advisor of my Ph.D. journey. She continually and convincingly conveyed a spirit of adventure in regard to research and scholarship, and an excitement in regard to teaching. Her wisdom, support and advices have guided me through all of my difficult moments, not only in doing research but also in my life. Without her guidance and persistent help this dissertation would not have been possible. Also, I am graceful to have excellent labmates who have provided extremely helpful resources to my study.

I would like to thank my committee members, Professor Sanjay Ranka, Professor Panos Pardalos, Professor Tamer Kahveci and Professor Prabhat Mishra who have been very supportive to my dissertation. Their encouragement and advices have helped me a lot not only in my Ph.D. study but also in my future career. Financial support for my Ph.D. program was provided by the University of Florida, NSF CAREER Award Grant number 0953284 and the DTRA Grant number HDTRA1-08-10.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

COMMUNITY STRUCTURE AND ITS APPLICATIONS
IN DYNAMIC COMPLEX NETWORKS

By

Nam P. Nguyen

May 2013

Chair: My T. Thai
Major: Computer Engineering

In this dissertation, we focus on analyzing and understanding the organizational principals, assessing the structural vulnerability as well as exploring practical applications of dynamic complex networks. In particular, we propose two adaptive frameworks for identifying the nonoverlapping and overlapping community structure in dynamic networks. Our approaches have not only the power of quickly and efficiently updating the network communities, but also the ability of tracing the evolution of those communities over time. We also suggest a detection method based on nonnegative matrix factorization which can work on weighted and directed networks. Consequently, we study the discovery of stable communities in the networks, i.e., communities which are tightly connected and remain wealthy even over a long period of time. Furthermore, we investigate on the structural vulnerability of the network community structure via identifying key nodes that play an important role in maintaining the normal function of the whole system. This is a new research direction on the cyber-infrastructure that we have recently introduced. To certify the effectiveness of our suggested frameworks and algorithms, we extensively test them on not only synthesized networks but also on real-world dynamic traces. Finally, we demonstrate the wide applicability of our algorithms via realistic applications, such as the limiting misinformation spread in Online Social Networks as well as the social-based forwarding and routing strategy and worm containment in Mobile networks.

CHAPTER 1
INTRODUCTION

1.1 Community Detection in Dynamic Complex Networks

Many complex systems in reality exhibit the property of containing community structure [37][85], i.e., they naturally divide into groups of vertices with denser connections inside each group and fewer connections crossing groups, where vertices and connections represent network users and their social interactions, respectively. Members in each community of a social network usually share things in common such as interests in photography, movies, music or discussion topics and thus, they tend to interact more frequently with each other than with members outside of their community. Community detection in a network is the gathering of network vertices into groups in such a way that nodes in each group are densely connected inside and sparser outside.

It is noteworthy to differentiate between community detection and graph clustering. These two problems share the same objective of partitioning network nodes into groups; however, the number of clusters is predefined or given as part of the input in graph clustering whereas the number of communities is typically unknown in community detection. Detecting communities in a network provides us meaningful insights to its internal structure as well as its organization principles. Furthermore, knowing the structure of network communities could also provide us more helpful points of view to some uncovered parts of the network, thus helps in preventing potential networking diseases such as virus or worm propagation. Studies on community detection on static networks can be found in an excellent survey [58] as well as in the work of [76][6][78][8] and references therein.

Real-world complex networks, however, are not always static. In fact, most of complex systems in reality (such as Facebook, Bebo and Twitter in OSNs) evolve and witness an expand in size and space as their users increase, thus lend themselves to the field of dynamic networks. A dynamic network is a special type of evolving complex

networks in which changes are frequently introduced over time. In the sense of an online social network, such as Facebook, Twitter or Flickr, changes are usually introduced by users joining in or withdrawing from one or more groups or communities, by friends and friends connecting together, or by new people making friend with each other. Any of these events seems to have a little effect to a local structure of the network on one hand; the dynamics of the network over a long period of time, on the other hand, may lead to a significant transformation of the network community structure, thus raises a natural need of reidentification. However, the rapidly and unpredictably changing topology of a dynamic social network makes it an extremely complicated yet challenging problem.

Although one can possibly run any of the static community detection methods, which are widely available [76][6][78][17], to find the new community structure whenever the network is updated, he may encounter some disadvantages that cannot be neglected: (1) the long running time of a specific static method on large networks (2) the trap of local optima and (3) the almost same reaction to a small change to some local part of the network. A better, much efficient and less time consuming way to accomplish this expensive task is to adaptively update the network communities from the previous known structures, which helps to avoid the hassle of recomputing from scratch. This adaptive approach is the main focus of our study in this paper. In Figure $1-1$, we briefly generalize the idea of dynamic network community structure adaptation. Here, the network evolves from time t to $t+1$ under the change ΔG_t. The adaptive algorithm $\mathcal{A}$ quickly finds the new community structure $\mathcal{C}(G_{t+1})$ based on the previous structure $\mathcal{C}(G_t)$ together with the changes ΔG_t.

1.2 Nonnegative Matrix Factorization for Community Detection

Community identification on complex networks is a well-established field and many efficient graph-based methods have been introduced in the literature (see [32] for an excellent survey). Unfortunately, these methods expose the strong dependence on some local parts of the network topology as well as the implicit meaning and interpretation

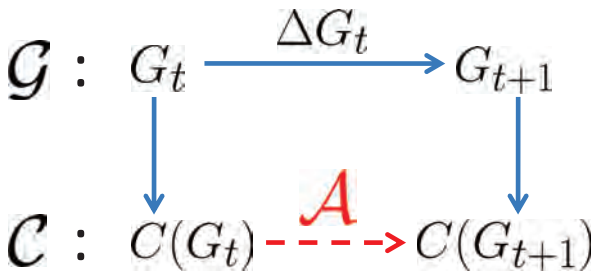


Figure 1-1. The general framework for our adaptive community detection algorithm $\mathcal{A}$.

from the detected overlapping communities. Recently, NMF-based algorithms for detecting network communities have gained great attention due to its meaningful interpretation [102]. In general, an NMF problem asks for, given a nonnegative matrix $X \in \mathbb{R}^{m\times m}$ and a number $k \ll \min\{m, n\}$, nonnegative matrices $W \in \mathbb{R}^{m\times k}$ and $H \in \mathbb{R}^{k\times n}$ such that $||X - WH||$ is minimized, where $||\cdot||$ is a cost function (usually the Frobenius distance or I-divergence). One notable property of NMF is its close relationship to K-mean clustering and graph partitioning [67][24], which also closely relates to community identification.

A few attempts have been suggested on this line of method. Lin et al proposed MetaFac [72], a NMF-based method for extracting community structure through relational hypergraphs. This method, however, is not capable for identifying overlapped structures. In [90], Prorakis et al. recently proposed an approach for finding overlapping communities using a Bayesian NMF based on hyperparameters. This method has the advantages of automatically determining the number of communities and not suffering from the resolution limit. Unfortunately, its built-in estimate of the number of communities could mislead the factorization to return a bad solution. In [103], Wang et al. proposed NMF methods on the Frobenius norm with the capability of extracting overlapped structures. However, we find these approaches do not appear to perform well on weighted directed networks as shown in the experiments.

To overcome the above limitations, we introduce two NMF approaches, namely iSNMF and iANMF, for effectively identifying social network communities with meaningful

interpretations. In particular, we are interested in approximating $X \approx HSH^{\top}$ since this factorization provides us H as the community indicator matrix and S as the community-interaction strength matrix, respectively. This factorization, as a result, nicely reflects the overlap of network communities and promises a meaningful community interpretation that is independent of the network topology.

1.3 Applications of The Network Community Structure

Detecting community structure of a dynamic social network is of considerable uses. To give a sense of it, consider the routing problem in communication network where nodes and links present people and mobile communications, respectively. Due to nodes mobility and unstable links properties of the network, designing an efficient routing scheme is extremely challenging. However, since people have a natural tendency to form groups of communication, there exist groups of nodes which are densely connected inside than outside in the underlying MANET as a reflection, and therefore, forms community structure in that MANET. An effective routing algorithm, as soon as it discovers the network community structure, can directly route or forward messages to nodes in the same (or to the related) community as the destination. By doing this way, we can avoid unnecessary messages forwarding through nodes in different communities, thus can lower down the number of duplicate messages as well as reduce the overhead information, which are essential in MANETs.

Another great example includes the worm containment in cellular networks [110], or in OSNs [81][82]. Nowadays, many social applications such as Facebook, Twitter and FourSquare, are able to run on open-API enabled mobile devices like PDAs and Iphones. However, if such an application is infected with malicious software, such as worms or viruses, this openness will also make it easier for their propagation. A possible solution to prevent worms from spreading out wider is to send patches to critical users and let them redistribute to the others. Intuitively, the smaller the set of important users for sending patches, the better. But how can we effectively choose that set of minimal

size? This is where community structure comes into the picture and helps. In particular, we show that selecting users in the boundaries of the overlapped nodes gives a tighter and more efficient set of influential users, thus significantly lowers the number of sent patches as well as overhead information, which are essential in cellular networks and OSNs.

1.4 The Identification of Stable Communities

OSNs in reality are highly dynamic as social interactions on them tend to come and go quickly. Consequently, their communities are also dynamical and evolve heavily as the networks change over time. However, Palla et al. observe in their seminal work that some communities in social networks are tightly connected and remain wealthy even over a long period of time [86]. The authors also point out that large-size communities with a high internal densities and less external distractions tend to remain stable during the network evolution, which intuitively agree with the findings reported in [49]. These observations reassemble the concept of stable communities in OSNs. For example, stable communities on Facebook can be visualized as groups of users who devoted themselves to one particular interest such as movie, music or photography. Likewise, a stable community in Twitter can be illustrated via a group of users who may follow many but only loyal to a specific celebrity. In a different perspective, stable communities in a citation network may refer to well-established research topics in the field whereas unstable communities may represent topical or recently arising research directions.

The discovery of these stable communities, as a consequence, will provide us valuable insights into the core properties and characteristics of not only each community but also of the network as a whole. This knowledge can further benefit information retrieval in OSNs as searches can be redirected to stable communities sharing the most similar characteristics to the queries for more meaningful answers. For instance, the search for well-established research topics in a citation network can be mined more effectively when one looks at its stable rather than unstable communities, as

discussed above. However, the large-scale and nonreciprocal topologies of OSNs in reality make the detection of stable community structure an extremely challenging yet topical problem.

1.5 The Assessment of Network Community Structure Vulnerability

Complex systems, despite their diversity in physical infrastructures and underlying interactions, expose to be extremely vulnerable under node attacks. In some scenarios, the failures of only a few key nodes are enough to bring the whole network operation down to its knees [25]. More importantly, this vulnerability can further be propagated to a wider population, leading to a much more devastating consequence. In order to develop a comprehensive understanding on this type of attack, it is therefore important to understand not only the impact of nodes' failures on the network components but also the inner and interdependency among those components [88]. Particularly, it is crucial to explore how the failure of a single node, or a set of nodes in general, can significantly change the structure of the network components as well as how these components would affect each other in cases of attacks. However, the large scale and dynamical properties of complex systems in practice make this a complicated problem.

To tackle this problem, we introduce the use of network modules to study both the impact of nodes's failures and the network component interdependency. There are several reasons and benefits behind this approach. First of all, investigating the interdependencies based on the topology of the underlying network structures is a major aspect that must be considered to understand the behavior of structural vulnerability [88]. Secondly, most complex networks commonly exhibit modular property, or in other words, they exhibit to contain community structure in their underlying organizations. That is, the network nodes can be gathered into groups in such a way that each group is densely connected internally and sparsely connected externally [38][75]. Nodes in each community usually share similar functions and characteristics that distinguish themselves from the others. In a broader view, communities displays the whole network

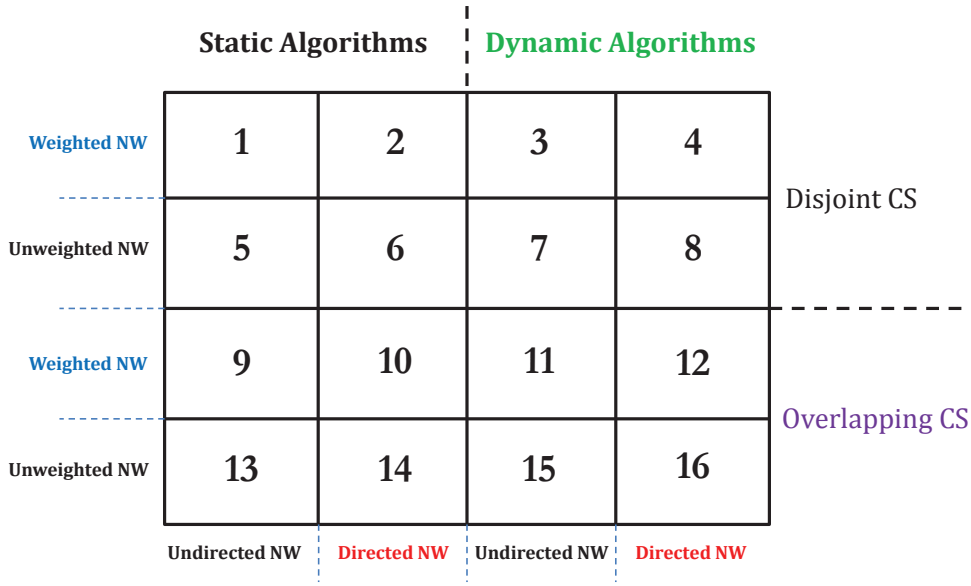


Figure 1-2. The classification of community detection algorithms in complex networks.

structure as a compact and more understandable level where a community may represent an entity or a functional group in the system. At this level, element failures in one community can have a profound impact which can consequently lead to changes of other communities. Therefore, identifying network elements that are essential to its community structure is a fundamental and extremely important issue. To the best of our knowledge, this research direction has not been addressed so far in the literature.

1.6 Literature Review

Community detection in dynamic networks

Community detection in complex networks is a well established field and a tremendous number of identification methods has been proposed in the literature. Some notable approach directions include classical graph clustering algorithms [4][73], dynamic approaches [92], modularity optimization methods [75], statistical inference [79] or random walk for community detection [28] (see [32] and references therein for an excellent survey).

In a general view, community structure detection algorithms for complex network can be classified in different ways: either by nonoverlapping or overlapping detection

algorithms, by static or dynamic algorithms, or by algorithms for directed and undirected networks, etc. Figure 1-2 describes a details classification of 16 different types of identification algorithms.

Community detection on static networks has attracted a lot of attentions and many efficient methods have been proposed for this type of networks. Detecting community structure on dynamic networks, however, has so far been an untrodden area. A recent work of Palla et al. [85] proposed an innovative method for detecting communities on dynamic networks based on the k-clique percolation technique. This approach can detect overlapping communities; however, it is time consuming, especially on large scale networks. Another recent work of Zhang et al. [109] proposed a detection method based on contradicting the network topology and the topology-based propinquity, where propinquity is the probability of a pair of nodes involved in a community. A work in [98] presented a parameter-free methodology for detecting clusters on time-evolving graphs based on mutual information and entropy functions of Information Theory. Hui et al. [48] proposed a distributed method for community detection in which modularity was used as a measure instead of objective function. A part from that, [44] attempted to track the evolving of communities over time, using a few static network snapshots.

In [99], the authors present a framework for identifying dynamic communities with a constant factor approximation. However, this method does not seem to make sense on real-world social networks since it requires some predefined penalty costs which are generally unknown on dynamic networks. A recent work [26], Thang et al. proposed a social-aware routing strategy in MANETs which also makes uses of a modularity-based procedure name MIEN for quickly updating the network structure. In particular, MIEN tries to compose and decompose network modules in order to keep up with the changes and uses fast modularity algorithm [76] to update the network modules. However, this method performs slowly on large scale dynamic networks due to the high complexity of [76].

In [70], Lin et al. proposed FacetNet, a framework for analyzing communities in dynamic networks based on the optimization of snapshot costs. FacetNet is guaranteed to converge to a local optimal solution; however, its convergence speed is slow and its input asks for the number of network communities which are usually unknown in practice. In [27], Duan et al. proposed Stream-Group, an incremental method to solve the community mining and detect the change points in weighted dynamic graphs. This method is modularity-based thus may inherit the resolution limit while discovering network communities. In another attempt, Kim et al. [52] suggested a particle-and-density based clustering method for dynamic networks, based on the extended modularity and the concepts of nano-community and /-quasi-clique-by-clique. Apart from that, the work of Cazabet et al. [9] proposed iLCD method to find the overlapping network communities by adding edges and then merging similar ones. However, this model might not be sufficient in consideration with the dynamic behaviors of the network when new nodes are introduced or removed, or when existing edges are removed from the network. In [60], the author presented OSLOM, a framework for testing the statistical significance of a cluster with respect to a global null model (e.g., a random graph). To expand a community, OSLOM locally computes the value r for each neighbor node and tries to include that node into the current community.

Nonnegative matrix factorization for community detection

Community detection on complex networks is a mature research area and besides NMF-based algorithms, many effective graph-based or topology-based algorithms have been proposed for this purpose. In general, detection methods can be classified into non-overlapping (disjoint) and overlapping algorithms. Traditional non-overlapping algorithms [75][17][77] may return good community identification results, however are not able to reveal the overlapped network structures, particularly on social networks. On the other category, algorithms for graph-based and topological-based detection of overlapping communities have also been proposed in the literature. Most of them

are based on the clique-percolation [84] or clique extension [63] techniques, on the extended modularity [62][83], on a specific fitness function [56], on label propagation [40], or link-based technique [1]. See [32] and references therein for an excellent survey on those detection methods.

Although the success of these aforementioned algorithms have been theoretically and empirically verified, they still expose the following limitations: (1) The strong dependence on some local parts of the network topology, e.g., the clique-percolation method depends on some dense subnetworks in order to percolate, a link-based technique relies on potential links with highest degrees, a modularity-based technique depends on the network hierarchy in order to maximize modularity, etc, and (2) The implicit meaning and interpretation from the detected overlapping communities, e.g., what is the contribution of an overlapped node to these percolated-cliques or why would it even be there? These shortcomings of these methods drive the need for a better approach with a more meaningful interpretation.

Stable community detection

The discovery of stable communities, on the contrary, is still an untrodden area with only a few attempts has been suggested [23][59][68]. This special property of network communities was perhaps first observed by Palla et al. in his seminal work [86], where they point out that tight-knit communities with high internal densities and less external distractions tend to remain strong over time, thereby reassembles the concept of community stability. Delvenne et al. [23] extend this general concept to proposed an measure, called "stability of the clustering $r(t, H)$", to quantify how stable a given cluster (or community structure) H is at a specific time step t based on the Markov Autocovariance model. Under this notation, a cluster H is stable at time t if a high value of $r(t, H)$ is observed. This quantity, instead, is more appropriate for verification rather than identification of stable network communities since it requires the specification of time step t a prior.

In a different approach, Lancichinetti et al. [59] investigate on the consensus of community detection methods. The authors report that, given a particular algorithm A, the consensus on communities found by A after multiple runs dramatically improve the quality of the detection, henceforth suggest that those communities are candidates for stable structures. This is a very interesting approach, however, might encounter some disadvantages of (1) the expensive computational cost and time consuming, and (2) the convergence of the whole iterative process is not guaranteed. In a recent attempt, Yanhua et al. [68] utilize the concept of mutual links and suggest an spectral-clustering-based identification method that tries to maximize the total mutual connections in order to find stable communities. However, there are possibilities that some mutual links are of low magnitudes, and thus, do not significantly contribute to the overall stability at the community level.

Structural vulnerability assessment of community structure

Community structure and complex network vulnerability are the two major and well-developed areas of networking research. Surveys on community structure detection algorithms as well as methods for assessing network vulnerabilities can be found in the work of Fortunatos et. al. [32], and Grubesic et. al. [41], respectively. However, assessing the vulnerability of network community structure has so far been an untrodden area. A large body of work has been devoted to find the node roles within a community by a link-based technique together with a modification of node degree [95], by using the spectrum of the graph [105], by using a within-module degree and their participation coefficient [42], or by the detection of key nodes, overlapping communities and “date” and “party” hubs [54]. However, none of these approaches discuss how the community structure would change in the failure of those important nodes, especially in terms of NMI measure.

The vulnerability of network function and structure has been examined under the node centrality metrics, such as high degree and betweeness centrality, as well as

under the average shortest path which tries to signify the lengths of shortest distances between node pairs [41], under the pairwise connectivity metric whose goal aims to break the network's pairwise connectivity down to a certain level [25], or under the available number of compromised $s - t$ flows [74], etc. However, there is an even more crucial risk that could dramatically affect the normal network functionality that has not been addressed so far: the transformation or restruction of the network community structure. Due to its vital role in the network, any significant restruction or transformation of the community structure, resulted from important node removal, can potentially change the entire network organization and consequently lead to a malfunction or unpredictable corruption of the whole network.

1.7 Dissertation outline

In chapter 2, we propose QCA, a fast and adaptive method for efficiently identifying the nonoverlapping community structure of a dynamic social network. Our approach takes into account the discovered structures and processes on network changes only, thus significantly reduces computational cost and processing time. We study the dynamics of a social network and prove theoretical results regarding its communities' behaviors over time, which are the bases of our method. We extensively evaluate our algorithms on both synthesized and real dynamic social traces. Experimental results show that QCA achieves not only competitive modularity scores but also high quality community structures in a timely manner. We apply QCA method to worm containment problem in OSNs. Simulation results show that QCA outperforms current available methods and confirm its applicability in social network problems.

In chapter 3, we suggest AFOCS, a two-phase adaptive framework for not only detecting and updating the overlapping network communities but also tracing their evolution over time. Theoretical analyses show AFOCS partially achieves more than 74% the internal density of the optimal solution. Second, we evaluate AFOCS on both synthesized and real traces in comparison to both the state-of-the-art and the

most popular static detection methods COPRA and CFinder, as well as to recent adaptive methods FacetNet, iLCD and OSLOM. Empirical results show that AFOCS achieves both competitively results and high quality community structures in a timely manner. Finally, with AFOCS, we suggest a community based forwarding strategy for communication networks that reduces up to 11x overhead information while maintaining competitively delivery time and ratio. We also propose a new social-aware patching scheme for containing worms in OSNs, which helps reducing up to 7x the infection rates on Facebook dataset.

We analyze two NMF approaches in chapter 4, namely iSNMF and iANMF, for effectively identifying social network communities with meaningful interpretations. In particular, we are interested in approximating $X \approx HSH^{\top}$ since this factorization provides us H as the foundation feature matrix and S as the feature interaction matrix. Alternatively, H and S can also be thought of as community indicator and inter-community strength matrices whose row elements can further be interpreted as probabilities of nodes belonging to different communities. This factorization, as a result, nicely reflects the overlap of network communities and promises a meaningful community interpretation that is independent of the network topology.

In an application perspective, we illustrate the practical applications of the network community structure via two emerging problems on social and mobile computing, namely the Worm spread containment problem on online social networks (chapter 5) and the forwarding and routing strategy (chapter 6) on mobile networks. We demonstrate that methods and strategies employing QCA and FOCS as community detection cores obtain a significant improvement in term of performance and solution quality. These realistic applications brighten the wide applicability of the network community structure many problems enabled my complex networks.

In chapter 7, we suggest an estimation which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD - a framework

to identify community structure in directional OSNs with the advantage of community stability. We next explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental mathematical theory to support the SCD framework. To certify the efficiency of our approach, we extensively test SCD on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT and NetHEPT_WC collaboration networks as well as Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

In chapter 8, we introduce CSV problem to assess the impact of nodes' failures on the network community structure. To the best of our knowledge, this is the first attempt in this line of research. We analyze possible conditions that can lead to the minimization of NMI on network community structures. We suggest the concept of generating edges of a community and provide an optimal solution for finding a MGES. We propose genEdge, a node selection strategy for CSV based on the MGES solution. We conducted experiments on both synthesized data with known community structures and real world traces. Empirical results reveal that genEdge outperforms other node selection strategies in terms of solution quality as well as in reference to different underlying community detection algorithms. In an application perspective, we demonstrate the critical importance of CSV via the forwarding and routing strategies in delay tolerant networks (DTNs), where the failures of some important devices significantly degrade the entire system's performance.

Finally, we summary our contributions and conclude the dissertation in chapter 9.

CHAPTER 2
NONOVERLAPPING COMMUNITY STRUCTURE DETECTION

In this chapter, we present QCA, our proposed algorithms for detecting nonoverlapping community structure in a dynamic complex network. In the following sections, we first introduce the preliminaries in section 2.1 and then describe our QCA method in detail in section 2.2. Finally, the empirical evaluations of QCA on both synthesized and real datasets are presented in section 2.3.

2.1 Problem Definition

We first present the notations, objective function as well as the dynamic graph model representing a social network that we will use throughout this section.

(Notation) Let $G = (V, E)$ be an undirected unweighted graph with N nodes and M links representing a social network. Let $\mathcal{C} = \{C_1, C_2, .., C_k\}$ denote a collection of disjoint communities, where $C_i \in \mathcal{C}$ is a community of G. For each vertex u, denote by d_u, $C(u)$ and $NC(u)$ its degree, the community containing u and the set of its adjacent communities. Furthermore, for any $S \subseteq V$, let m_S, d_S and e_S^u be the number of links inside S, the total degree of vertices in S and the number of connections from u to S, respectively. The pairs of terms community and module; node and vertex as well as edge and link and are used interchangeably.

(Dynamic social network) Let $G^s = (V^s, E^s)$ be a time dependent network snapshot recorded at time s. Denote by ΔV^s and ΔE^s the sets of vertices and links to be introduced (or removed) at time s and let $\Delta G^s = (\Delta V^s, \Delta E^s)$ denote the change in term of the whole network. The next network snapshot G^{s+1} is the current one together with changes, i.e., $G^{s+1} = G^s \cup \Delta G^s$. A dynamic network $\mathcal{G}$ is a sequence of network snapshots evolving over time: $\mathcal{G} = (G^0, G^1, .., G^s)$.

(Objective function) In order to quantify the goodness of a network community structure, we take into account the most widely accepted measure called modularity Q

[78], which is defined as:

$$Q = \sum_{C \in \mathcal{C}} \left(\frac{m_C}{M} - \frac{d_C^2}{4M^2}\right).$$

Basically, Q is the fraction of all links within communities subtracts the expected value of the same quantity in a graph whose nodes have the same degrees but links are distributed randomly, and the higher modularity Q, the better network community structure is. Therefore, our objective is to find a community assignment for each vertex in the network such that Q is maximized. Modularity, just like other quality measurements for community identifications, has some certain disadvantages such as its non-locality and scaling behavior [8], or resolution limit [35]. However, it is still very well considered due to its robustness and usefulness that closely agree with intuition on a wide range of real world networks.

Problem Definition: Given a dynamic social network $\mathcal{G} = (G^0, G^1, .., G^s)$ where G^0 is the original network and G^1, G^2,.., G^s are the network snapshots obtained through ΔG^1, ΔG^2,.., ΔG^s, we need to devise an adaptive algorithm to efficiently detect and identify the network community structure at any time point utilizing the information from the previous snapshots as well as tracing the evolution of the network community structure.

2.2 Algorithm Description

Let us first discuss how changes to the evolving network topology affect the structure of its communities. We use the term intra-community links to denote edges whose two endpoints belong to the same community, and the term inter-community links to denote those with endpoints connecting different communities. For each community C, the connections linking C with other communities are much fewer than those within C itself, i.e., nodes in C are densely connected inside and less densely connected outside. Intuitively, adding intra-community links inside or removing inter-community links between communities of G will strengthen those communities and make the structure of G more clear. Vice versa, removing intra-community links and inserting inter-community links will loosen the structure of G. The community updating process, as a result, is

challenging since an insignificant change in the network topology can possibly lead to an unexpected transformation of its community structure.

We will discuss in detail possible behaviors of dynamic network communities in Figure 2-1. 2-1A: New edge (u, v): u and v are first checked and memberships are then tested on X and Y. 2-1B: (a) The original community (b) After the dotted edge is removed, two smaller communities arise. 2-1C: (a) The original four communities (b) After the central node is removed, the leftover nodes join in different modules, forming three new communities. 2-1D: (a) The original community (b) When g is removed, a 3-clique is placed at a to discover b, c, d and e. f assigned singleton afterwards.

In order to reflect changes introduced to the social network, its underlying graph is constantly updated by either inserting or removing a node or a set of nodes, or by either introducing or deleting an edge or a set of edges. In fact, the introduction or removal of a set of nodes (or edges) can be decomposed as a sequence of node (or edge) insertions (or removals), in which a single node (or a single edge) is introduced (or removed) at a time. This observation helps us to treat network changes as a collection of simple events where a simple event can be one of newNode, removeNode, newEdge, removeEdge whose details are as follow:

- newNode $(V \cup \{u\})$: A new node u with its associated edges are introduced. u could come with no or more than one new edge(s).
- removeNode $(V \backslash \{u\})$: A node u and its adjacent edges are removed from the network.
- newEdge $(E \cup \{e\})$: A new edge e connecting two existing nodes is introduced.
- removeEdge $(E \backslash \{e\})$: An existing edge e in the network is removed.

Our approach first requires an initial community structure $\mathcal{C}_0$, which we call the basic structure, in order to process further. Since the input model is restricted as an undirected unweighted network, this initial community structure can be obtained by performing any of the available static community detection methods [76][6][17]. To

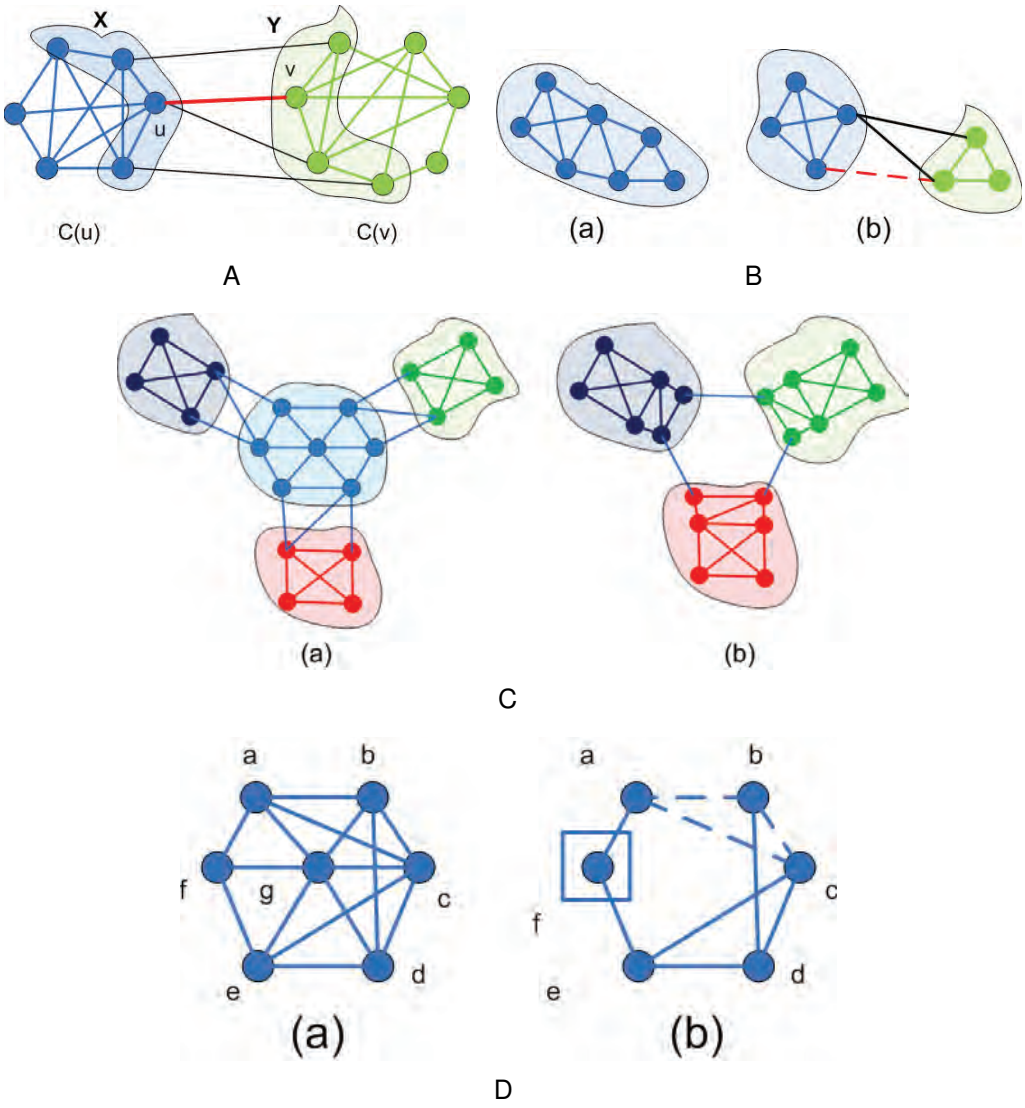


Figure 2-1. Possible behaviors of the network community structure during evolution.

obtain a good basic structure, we choose the method proposed by Blondel et al. in [6] which produces a good network community structure in a timely manner [58].

2.2.1 New node

Let us consider the first case when a new node u and its associated connections are introduced. Note that u may come with no adjacent edges or with many of them connecting one or more communities. If u has no adjacent edge, we create a new community for it and leave the current structure intact. The interesting case happens, and it usually does, when u comes with edges connecting one or more existing

communities. In this latter situation, we need to determine which community u should join in in order to maximize the gained modularity. There are several local methods introduced for this task, for instance the algorithms of [76][17]. Our method is inspired by a physical approach proposed in [107], in which each node is influenced by two forces: F_{in}^{C} (to keep u stays inside community C) and F_{out}^{C} (the force a community C makes in order to bring u to C) defined as follow:

$$F_{in}^{C}(u) = e_{C}^{u} - \frac{d_u(d_C - d_u)}{2M},$$

and

$$F_{out}^{S}(u) = \max_{S \in NC(u)} \{e_{S}^{u} - \frac{d_u d_{outS}}{2M}\},$$

where d_{outS} is of opposite meaning of d_S.

Taking into account the above two forces, a node v can actively determines its best community membership by computing those forces and either lets itself join the community S having the highest $F_{out}^{S}(v)$ (if $F_{out}^{S}(v) > F_{in}^{C(v)}(v)$) or stays in the current community $C(v)$ otherwise. By Theorem 2.1, we bridge the connection between those forces and the objective function, i.e., joining the new node in the community with the highest outer force will maximize the local gained modularity. The process is presented in Alg. 1.

Theorem 2.1. *Let C be the community having the maximum $F_{out}^{C}(u)$ when a new node u with degree p is added to G, then joining u in C gives the maximal gained modularity.*

Proof. Let D be a community of G and $D \neq C$, we show that joining u in D contributes less modularity than joining u in C. The overall modularity Q when u joins in C is

$$Q_1 = \frac{m_C + e_{C}^{u}}{M + p} - \frac{(d_C + e_{C}^{u} + p)^2}{4(M + p)^2} + \frac{m_D}{M + p} - \frac{(d_D + e_{D}^{u})^2}{4(M + p)^2} + A,$$

Algorithm 1 New_Node

Input: New node u with associated links; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$

Create a new community of only u;
for $v \in N(u)$ **do**
    Let v determine its best community;
end for
for $C \in NC(u)$ **do**
    Find $F_{out}^{C}(u)$;
end for
if $\max_C F_{out}^{C}(u) > F_{in}^{C_u}(u)$ **then**
    Let $C_u \leftarrow \arg\max_C \{F_{out}^{C}(u)\}$;
    Update $\mathcal{C}_{t+1} : \mathcal{C}_{t+1} \leftarrow \big(\mathcal{C}_t \backslash C_u\big) \cup \big(C_u \cup u\big)$;
end if

where A is the summation of other modularity contributions. Similarly, joining u to D gives

$$Q_2 = \frac{m_C}{M+p} - \frac{(d_C + e_C^u)^2}{4(M+p)^2} + \frac{m_D + e_D^u}{M+p} - \frac{(d_D + e_D^u + p)^2}{4(M+p)^2} + A,$$

and

$$Q_1 - Q_2 = \frac{1}{M+p}\big(e_C^u - e_D^u + \frac{p(d_D - d_C + e_D^u - e_C^u)}{2(M+p)}\big).$$

Now, since C is the community that gives the maximum $F_{out}^{C}(u)$, we obtain

$$e_C^u - \frac{p(d_C + e_C^u)}{2(M+p)} > e_D^u - \frac{p(d_D + e_D^u)}{2(M+p)},$$

which implies

$$e_C^u - e_D^u + \frac{p(d_D - d_C + e_D^u - e_C^u)}{2(M+p)} > 0.$$

Hence, $Q_1 - Q_2 > 0$ and thus the conclusion follows. □

2.2.2 New edge

When a new edge $e = (u, v)$ connecting two existing vertices u, v is introduced, we divide it further into two subcases: e is an intra-community link (totally inside a community C) or an inter-community link (connects two communities $C(u)$ and $C(v)$). If e is inside a community C, its presence will strengthen the inner structure of C

according to Lemma 1. Furthermore, by Lemma 2, we know that adding e should not split the current community C into smaller modules. Therefore, we leave the current network structure intact in this case.

The interesting situation occurs when e is a link connecting communities $C(u)$ and $C(v)$ since its presence could possibly make u (or v) leave its current module and join in the new community. Additionally, if u (or v) decides to change its membership, it can advertise its new community to all its neighbors and some of them might eventually want to change their memberships as a consequence. By Lemma 3, we show that should u (or v) ever change its community assignment, $C(v)$ (or $C(u)$) is the best new community for it. But how can we quickly decide whether u (or v) should change its membership in order to form a better community structure with higher modularity? To this end, we provide a criterion to test for membership changing of u and v in Theorem 2.2. Here, if both $\Delta q_{u,C,D}$ and $\Delta q_{v,C,D}$ fail to satisfy the criteria, we can safely preserve the current network community structure (Corollary 1). Otherwise, we move u (or v) to its new community and consequently let its neighbors determine their best modules to join in, using local search and swapping to maximize gained modularity. Figure 2-1A describes the procedure for this latter case. The detailed algorithm is described in Alg. 2.

Lemma 1. *For any $C \in \mathcal{C}$, if $d_C \leq M - 1$ then adding an edge within C will increase its modularity contribution.*

Proof. The portion Q_1 that community C contributes to the overall modularity Q is

$$Q_{1C} = \frac{m_C}{M} - \frac{d_C^2}{4M^2}.$$

When a new edge coming in, the new modularity Q_2 is

$$Q_{2C} = \frac{m_C + 1}{M + 1} - \frac{(d_C + 2)^2}{4(M + 1)^2}.$$

Algorithm 2 New_Edge

Input: Edge $\{u, v\}$ to be added; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

if (u and $v \notin V$) **then**
    $\mathcal{C}_{t+1} \leftarrow \mathcal{C}_t \cup \{u, v\}$;
else if $C(u) \neq C(v)$ **then**
    if $\Delta q_{u,C(u),C(v)} < 0$ and $\Delta q_{v,C(u),C(v)} < 0$ **then**
        return $\mathcal{C}_{t+1} \equiv \mathcal{C}_t$;
    else
        $w = \arg\max\{\Delta q_{u,C(u),C(v)}, \Delta q_{v,C(u),C(v)}\}$;
        Move w to the new community;
        for $t \in N(w)$ **do**
            Let t determine its best community;
        end for
        Update C_{t+1};
    end if
end if

Now, taking the difference between Q_2 and Q_1 gives

$$\begin{aligned}
\Delta Q_C &= Q_{2C} - Q_{1C} \\
&= \frac{4M^3 - 4m_C M^2 - 4d_C M^2 - 4m_C M + 2d_C^2 M + d_C^2}{4(M+1)^2 M^2} \\
&\geq \frac{4M^3 - 6d_C M^2 - 2d_C M + 2d_C^2 M + d_C^2}{4(M+1)^2 M^2} \quad (\text{since } m_C \leq \tfrac{d_C}{2}) \\
&\geq \frac{(2M^2 - 2d_C M - d_C)(2M - d_C)}{4(M+1)^2 M^2} \geq 0
\end{aligned}$$

The last inequality holds since $d_C \leq M - 1$ implies $2M^2 - 2d_C M - d_C \geq 0$. □

Lemma 2. *If C is a community in the current snapshot of G, then adding any intra-community link to C should not split it into smaller modules.*

Proof. Assume the contradiction, i.e, C should be divided into smaller modules when an edge is added into it. Let $X_1, X_2, .., X_k$ be disjoint subsets of C representing these modules. Let d_i and e_{ij} be the total degree of vertices inside X_i and the number of links going from X_i to X_j, respectedly. Assume that, W.L.O.G., when an edge is added inside

C, it is added to X_1. We will show that

$$\frac{\sum_{i\neq j} d_i d_j}{2M} < \sum_{i\neq j} e_{ij} < \frac{\sum_{i\neq j} d_i d_j}{2M} + 1,$$

which can not happen since $\sum_{i\neq j} e_{ij}$ is an natural number. Recall that

$$Q_{1C} = \frac{m_C}{M} - \frac{d_C^2}{4M^2},$$

and

$$Q_{X_i} = \frac{m_i}{M} - \frac{d_i^2}{4M^2},$$

and prior to adding an edge to C, we have

$$Q_{1C} > \sum_{i=1}^{k} Q_{X_i},$$

or equivalently,

$$\frac{m_C}{M} - \frac{d_C^2}{4M^2} > \sum_{i=1}^{k} \left(\frac{m_i}{M} - \frac{d_i^2}{4M^2}\right).$$

Since $X_1, X_2, .., X_k$ are disjoint subsets of C, it follows that

$$d_C = \sum_{i=1}^{k} d_i$$

and

$$m_C = \sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij},$$

(where m_i is the number of links inside X_i). The above inequality equals to

$$\frac{m_C}{M} - \sum_{i=1}^{k} \frac{m_i}{M} > \frac{d_C^2}{4M^2} - \sum_{i=1}^{k} \frac{d_i^2}{4M^2},$$

or

$$\sum_{i<j} e_{ij} > \left\lceil \frac{\sum_{i<j} d_i d_j}{2M} \right\rceil.$$

Now, assume that the new edge is added to X_1 and C is split into $X_1, X_2, .., X_k$ which implies that dividing C into k smaller communities will increase the overall modularity,

i.e.,

$$Q_{2C} < \sum_{i=1}^{k} Q_{2X_i}.$$

Now,

$$\begin{aligned}
&Q_{2C} < \sum_{i=1}^{k} Q_{2X_i} \\
&\Leftrightarrow \frac{\sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^{k} d_i + 2\right)^2}{4(M+1^2)} < \frac{m_1+1}{M+1} - \frac{(d_1+2)^2}{4(M+1)^2} + \sum_{i=2}^{k} \left(\frac{m_i}{M+1} - \frac{d_i^2}{4(M+1)^2}\right) \\
&\Leftrightarrow \quad \frac{\sum_{i=1}^{k} m_i + \sum_{i<j} e_{ij} + 1}{M+1} - \frac{\left(\sum_{i=1}^{k} d_i + 2\right)^2}{4(M+1^2)} < \frac{\sum_{i=1}^{k} m_i + 1}{M+1} - \frac{(d_1+2)^2}{4(M+1)^2} - \sum_{i=2}^{k} \frac{d_i^2}{4(M+1)^2} \\
&\Leftrightarrow \quad \sum_{i<j} e_{ij} < \frac{\sum_{i=1}^{k} d_i - 2d_1 + \sum_{i<j} d_i d_j}{2(M+1)}
\end{aligned}$$

Moreover, since it is obvious that $\sum_{i=1}^{k} d_i - 2d_1 < 2M$, we have

$$\frac{\sum_{i=1}^{k} d_i - 2d_1 + \sum_{i\neq j} d_i d_j}{2(M+1)} < \left\lceil \frac{\sum_{i<j} d_i d_j}{2M} \right\rceil + 1,$$

and thus the conclusion follows. □

Lemma 3. *When a new edge (u, v) connecting communities $C(u)$ and $C(v)$ is introduced, $C(v)$ (or $C(u)$) is the best candidate for u (or v) if it should ever change its membership.*

Proof. Let $C \equiv C(u)$ and $D \equiv C(v)$. Recall the outer force that a community S applies to vertex u is

$$F_{out}^{S}(u) = e_u^S - \frac{d_u d_{outS}}{2M}.$$

We will show that the presence of edge (u, v) will strengthen $F_{out}^{D}(u)$ while weakening the other outer forces $F_{out}^{S}(u)$, i.e, we show that $F_{out}^{D}(u)$ increases while $F_{out}^{S}(u)$

decreases for all $S \notin \{C, D\}$.

$$\begin{aligned} F^D_{out}(u)_{new} - F^D_{out}(u)_{old} &= \big(e^D_u + 1 - \frac{(d_u+1)(d_{outD}+1)}{2(M+1)}\big) - \big(e^D_u - \frac{d_u d_{outD}}{2M}\big) \\ &= \frac{2M + d_u d_{outD}}{2M} - \frac{d_u d_{outD} + d_{outD} + d_u + 1}{2(M+1)} \\ &\geq \frac{2M + d_u d_{outD}}{2(M+1)} - \frac{d_u d_{outD} + d_{outD} + d_u + 1}{2(M+1)} > 0 \end{aligned}$$

and thus $F^D_{out}(u)$ is strengthened when (u, v) is introduced. Furthermore, for any community $S \in \mathcal{C}$ and $S \notin \{C, D\}$, we have

$$\begin{aligned} F^S_{out}(u)_{new} - F^S_{out}(u)_{old} &= \big(e^S_u - \frac{(d_u+1)d_{outS}}{2(M+1)}\big) - \big(e^S_u - \frac{d_u d_{outS}}{2M}\big) \\ &= d_{outS}\big(\frac{d_u}{2M} - \frac{d_u+1}{2(M+1)}\big) < 0 \end{aligned}$$

which implies $F^S_{out}(u)$ is weakened when (u, v) is connected. Hence, the conclusion follows. □

Theorem 2.2. *Assume that a new edge (u, v) is added to the network. Let $C \equiv C(u)$ and $D \equiv C(v)$. If*

$$\Delta q_{u,C,D} \equiv 4(M+1)(e^u_D + 1 - e^u_C) + e^u_C(2d_D - 2d_u - e^u_C) - 2(d_u+1)(d_u + 1 + d_D - d_C) > 0$$

then joining u to D will increase the overall modularity.

Proof. Node u should leave its current community C and join in D if

$$Q_{D+u} + Q_{C-u} > Q_C + Q_D,$$

or equivalently,

$$\begin{aligned} &\frac{m_D + e_D + 1}{M+1} - \frac{(d_D + d_u + 2)^2}{4(M+1)^2} + \frac{m_C - e_C}{M+1} - \frac{(d_C - d_u - e_C)^2}{4(M+1)^2} \\ &\qquad > \frac{m_D}{M+1} - \frac{(d_D+1)^2}{4(M+1)^2} + \frac{m_C}{M+1} - \frac{(d_C+1)^2}{4(M+1)^2} \end{aligned}$$

The above equation equals to

$$4(M+1)(e_D+1-e_C)+e_C(2d_D-2d_u-e_C)-2(d_u+1)(d_u+1+d_D-d_C)>0,$$

which concludes the Theorem. □

Corollary 1. *If the condition in Theorem 2.2 is not satisfied, then neither u nor its neighbors should be moved to D.*

Proof. The proof follows from Theorem 2.2. □

2.2.3 Node removal

When an existing node u in a community C is removed, all of its adjacent edges are disregarded as a result. This case is challenging in the sense that the resulting community is very complicated: it can be either unchanged or broken into smaller pieces and could probably be merged with other communities. Let's consider two extreme cases when a single degree node and a node with highest degree in a community is removed. If a single degree node is removed, it leaves the resulted community unchanged (Lemma 5). However, when a highest degree vertex is removed, the current community might be disconnected and broken in to smaller pieces which then are merged to other communities as depicted in Figure 2-1C. Therefore, identifying the leftover structure of C is a crucial part once a vertex in C is removed.

To quickly and efficiently handle this task, we utilize the clique percolation method presented in [85]. In particular, when a vertex u is removed from C, we place a 3-clique to one of its neighbors and let the clique percolate until no vertices in C are discovered (Figure 2-1D). We then let the remaining communities of C choose their best communities to merge in. The detailed algorithm is presented in Alg. 3.

2.2.4 Edge removal

In the last case when an edge $e = (u, v)$ is removed, we divide further into four subcases (1) e is a single edge connecting only u and v (2) either u or v has

Algorithm 3 Node_Removal

Input: Node $u \in C$ to be removed; Current structure $\mathcal{C}_t$.
Output: An updated structure $\mathcal{C}_{t+1}$.

$i \leftarrow 1$;
while $N(u) \neq \emptyset$ **do**
    $S_i = \{$Nodes found by a 3-clique percolation on $v \in N(u)\}$;
    if $S_i == \emptyset$ **then**
        $S_i \leftarrow \{v\}$;
    end if
    $N(u) \leftarrow N(u) \backslash S_i$;
    $i \leftarrow i + 1$;
end while
Let each singleton in $N(u)$ consider its best communities;
Let each S_i consider its best communities as in [6]
Update $\mathcal{C}_t$;

degree one (3) e is an inter-community link connecting $C(u)$ and $C(v)$ and (4) e is an intra-community link. If e is an single edge, its removal will result in the same community structure plus two singletons of u and v themselves. The same reaction applies to the second subcase when either u or v has single degree due to Lemma 5, thus results in the prior network structure plus u (or v). When e is an inter-community link, the removal of e will strengthen the current network communities (Lemma 4) and hence, we just make no change to the overall network structure.

The last but most complicated case happens when an intra-community link is deleted. As depicted in Figure 2-1B, removing this kind of edge often leaves the community unchanged if the community itself is densely connected; however, the target module will be divided if it contains substructures which are less attractive or loosely connected to each other. Therefore, the problem of identifying the structure of the remaining modules is important. Theorem 2.3 provides us a convenient tool to test for community bi-division when an intra-community link is removed from the host community C. However, it requires an intensive look for all subsets of C, which may be time consuming when C is big. Note that prior to the removal of (u, v), the community C hosting this link should contain dense connections within itself and thus, the removal

of (u, v) should leave some sort of 'quasi-clique' structure [85] inside C. Therefore, we find all maximal quasi-cliques within the current community and have them (as well as leftover singletons) determine their best communities to join in. The detailed procedure is described in Alg. 4.

Algorithm 4 Edge_Removal

Input: Edge (u, v) to be removed; Current structure $\mathcal{C}_t$.
Output: An updated clustering $\mathcal{C}_{t+1}$.

if (u, v) is a single edge **then**
    $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if Either u (or v) is of degree one **then**
    $\mathcal{C}_{t+1} = (\mathcal{C}_t \backslash C(u)) \cup \{u\} \cup \{C(u) \backslash u\}$;
else if $C(u) \neq C(v)$ **then**
    $\mathcal{C}_{t+1} = \mathcal{C}_t$;
else
    % Now (u, v) is inside a community C %
    L = {Maximal quasi-cliques in C};
    Let the singletons in $C \backslash L$ consider their best communities;
end if
Update $\mathcal{C}_{t+1}$;

Lemma 4. *If C and D are two communities of G, then the removal of an inter-community link connecting them will strengthen modularity contributions of both C and D.*

Proof. Let Q_{1C} (resp. Q_{1D}) and Q_{2C} (resp. Q_{2D}) be the modularities of C (resp. D) before and after the removal of that link. We show that $Q_{2C} > Q_{1C}$ (and similarly, $Q_{2D} > Q_{1D}$) and thus, C and D contribute higher modularities to the network.

$$\begin{aligned} Q_{2C} - Q_{1C} &= \left(\frac{m_1}{M-1} - \frac{(d_1-1)^2}{4(M-1)^2}\right) - \left(\frac{m_1}{M} - \frac{d_1^2}{4M^2}\right) \\ &= m_1\left(\frac{1}{M-1} - \frac{1}{M}\right) + \frac{1}{4}\left(\frac{d_1}{M} - \frac{d_1-1}{M-1}\right)\left(\frac{d_1}{M} + \frac{d_1-1}{M-1}\right) \end{aligned}$$

Since all terms are all positive, $Q_{2C} - Q_{1C} > 0$. The same technique applies to show that $Q_{2D} > Q_{1D}$. □

Lemma 5. *The removal of (u, v) inside a community C where only u or v is of degree one will not separate C.*

Proof. Assume the contradiction, i.e., after the removal of (u, v) where $d_u = 1$, C is broken into smaller communities X_1, X_2,..., X_k which contribute higher modularity: $Q_{X_1} + \ldots + Q_{X_k} > Q_C$. W.L.O.G., suppose u was connected to X_1 prior to its removal. It follows that $Q_{X_1+u} > Q_{X_1}$ and thus $Q_{X_1+u} + \ldots + Q_{X_k} > Q_C$, which raises a contradiction since C is originally a community of $\mathcal{C}$. □

Lemma 6. *(Separation of a community) Let $C_1 \subseteq C$ and $C_2 = C \backslash C_1$ be two disjoint subsets of C. $(\mathcal{C} \backslash C) \cup \{C_1, C_2\}$ is a community structure with higher modularity when an edge crossing C_1 and C_2 is removed, i.e., C should be separated into C_1 and C_2, if and only if $e_{12} < \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1$.*

Proof. Let q_1, q_2 and q_C denote the modularity contribution of C_1, C_2 and C after an edge crossing (C_1, C_2) has been removed. Now,

$$
\begin{aligned}
e_{12} < \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1 &\Leftrightarrow \frac{2d_1 d_2 - 2d_C + 2}{4(M-1)^2} > \frac{e_{12} - 1}{M-1} \\
&\Leftrightarrow \frac{(d_1 + d_2 - 2)^2}{4(M-1)^2} - \frac{(d_1 - 1)^2}{4(M-1)^2} - \frac{(d_2 - 1)^2}{4(M-1)^2} \\
&\quad > \frac{m_1 + m_2 + e_{12} - 1}{M-1} - \frac{m_1 - 1}{M-1} - \frac{m_2 - 1}{M-1} \\
&\Leftrightarrow \frac{m_1 - 1}{M-1} - \frac{(d_1 - 1)^2}{4(M-1)^2} + \frac{m_2 - 1}{M} - \frac{(d_2 - 1)^2}{4(M-1)^2} \\
&\quad > \frac{m_1 + m_2 + e_{12} - 1}{M-1} - \frac{(d_1 + d_2 - 2)^2}{4(M-1)^2} \\
&\Leftrightarrow q_1 + q_2 > q_C.
\end{aligned}
$$

Thus, the conclusion follows. □

Theorem 2.3. *(Community bi-division) For any community C, let α and β be the lowest and the second highest degree of vertices in C, respectively. Assume that an edge e is removed from C. If there do not exist subsets $C_1 \subseteq C$ and $C_2 \equiv C \backslash C_1$ such that e is crossing C_1 and C_2 and $\frac{\min\{\alpha(d_C - \alpha), \beta(d_C - \beta)\}}{2M} < e_{12} < \frac{(d_C - 2)^2}{8(M-1)} + 1$, then any bi-division of C will not benefit the overall Q.*

Proof. From Lemma 6, it follows that in order to really benefit the overall modularity we must have

$$\frac{d_1 d_2}{2M} < e_{12} < \frac{d_1 d_2 + 1}{2(M-1)} + 1.$$

Now we find an upper bound for the RHS inequality. Since $d_1 + d_2 = d_C$, it follows that

$$\begin{aligned} e_{12} &< \frac{d_1 d_2 - d_C + 1}{2(M-1)} + 1 \leq \frac{\frac{(d_1+d_2)^2}{4} - d_C + 1}{2(M-1)} + 1 \\ &\leq \frac{\frac{d_C^2}{4} - d_C + 1}{2(M-1)} + 1 = \frac{(d_C - 2)^2}{8(M-1)} + 1 \end{aligned}$$

For a lower bound of the LHS inequality, we rewrite $d_1 d_2$ as $d_1 d_2 = d_1(d_C - d_1) = d_1 d_C - d_1^2$ and find the non-zero minimum value on the range $d_1 \in [\alpha, \beta]$. In this interval, $d_1 d_C - d_1^2$ is minimized either at $d_1 = \alpha$ or $d_1 = \beta$. Therefore,

$$\frac{\min\{\alpha(d_C - \alpha), \beta(d_C - \beta)\}}{2M} \leq \frac{d_1 d_2}{2M} < e_{12} \leq \frac{(d_C - 2)^2}{8(M-1)} + 1$$

□

Finally, our QCA method for quickly updating the network community structure is presented in Alg. 5.

Algorithm 5 **Q**uick **C**ommunity **A**daptation (QCA)

Input: $G \equiv G_0 = (V_0, E_0)$, $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, .., \mathcal{E}_s\}$ a collection of simple events
Output: Community structure $\mathcal{C}_t$ of G^t at time t.

```
Use [6] to find an initial community clustering C_0 of G_0;
for (t ← 1 to s) do
    C_t ← C_{t−1};
    if E_t = newNode(u) then
        New_Node(C_t, u);
    else if E_t = newEdge((u, v)) then
        New_Edge(C_t, (u, v));
    else if E_t = removeNode(u) then
        Remove_Node(C_t, u);
    else
        Remove_Edge(C_t, (u, v));
    end if
end for
```

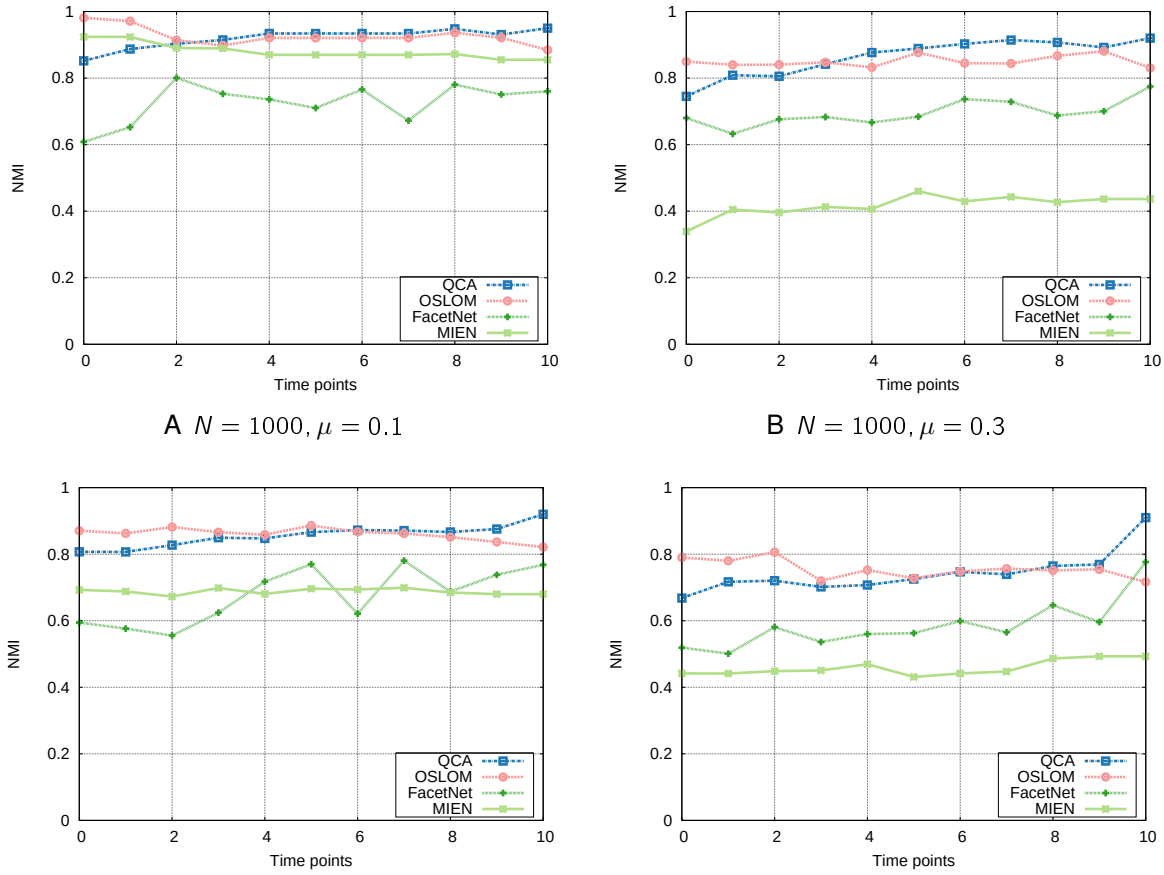
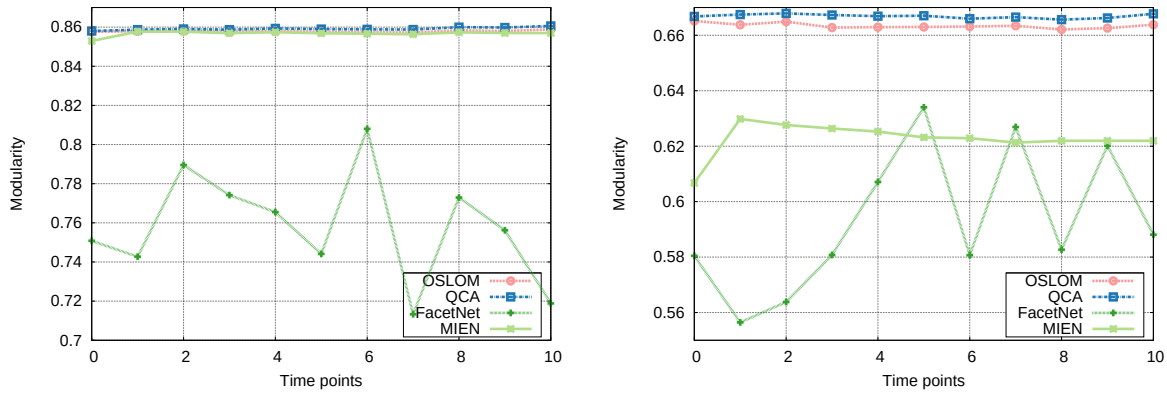


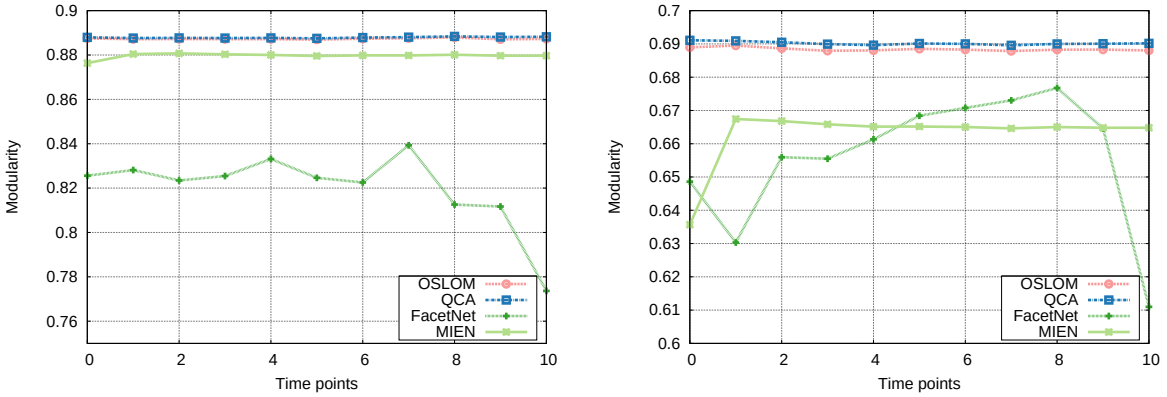
Figure 2-2. NMI scores on synthesized networks with known communities

2.3 Experimental Results

In this section, we first validate our approaches on different synthesized networks with known groundtruths, and then present our findings on real world traces including the Enron email [98], arXiv eprint citation [22], and Facebook social networks [100]. To certify the performance of our algorithms, we compare QCA to other adaptive community detection methods including (1) MIEN algorithm proposed by Thang et al. [26], (2) FacetNet framework proposed by Lin et al. [70], and (3) OSLOM method suggested by Lancichinetti et al. [60].



A $N = 1000, \mu = 0.1$ B $N = 1000, \mu = 0.3$



C $N = 5000, \mu = 0.1$ D $N = 5000, \mu = 0.3$

Figure 2-3. Modularity values on synthesized networks with known communities

2.3.1 Results on synthesized networks

Of course, the best way to evaluate our approaches is to validate them on real networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topology. Although synthesized networks might not reflect all the statistical properties of real ones, they provide us known ground truths via planted communities, and the ability to vary other parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has become an usual practice that is widely accepted in the field [58]. Hence, comparing QCA with other dynamic methods

on synthesized networks not only certifies its performance but also provides us the confidence to its behaviors on real world traces.

Setup. We use the well-known LFR benchmark [58] to generate 40 networks with 10 snapshots. Parameters are: the number of nodes $N = \{1000, 5000\}$, the mixing parameter $\mu = \{0.1, 0.3\}$ controlling the overall sharpness of the community structure. In order to quantify the similarity between the identified communities and the ground truth, we adopt a well known measure in Information Theory called Normalized Mutual Information (NMI). NMI has been proven to be reliable and is currently used in testing community detection algorithms [58]. Basically, $NMI(U, V)$ equals 1 if structures U and V are identical and equals 0 if they are totally separated, and the higher NMI the better.

Results. The NMI and Modularity values are reported in Figures 2-2 and 2-3. As depicted in their subfigures, the NMI values and modularities indicated by our QCA method, in general, are very high and competitive with those of OSLOM while are much better than those produced by MIEN and FacetNet methods. On these generated networks, we observe that MIEN and FacetNet perform well when the mixing parameter μ is small, i.e., when the network community structures are clear, however, their performances degrade dramatically when these structures become less clear as μ gets larger. Particularly, MIEN' and FacetNet' NMI scores and modularities in all test cases are fairly low and usually from 10% to 50% and 5% to 15% worst than those produced by QCA. This implies the network communities revealed by these methods are not as high similarity to the ground-truth as QCA algorithm. On the generated networks, OSLOM algorithm performs very well as suggested through its high NMI scores and modularity values. In particular, OSLOM tends to perform better than QCA in the first couple of network snapshots, however, its performance is taken over by QCA when the networks evolve over time, especially at the end of the evolution where OSLM reveals big gaps in similarity to the planted network communities (Note that the higher NMI score at the end of the evolution, the better the final detected community structure). This

concludes that the network communities discovered by QCA are of the best similarity to ones planted in the ground-truth in comparison with other methods.

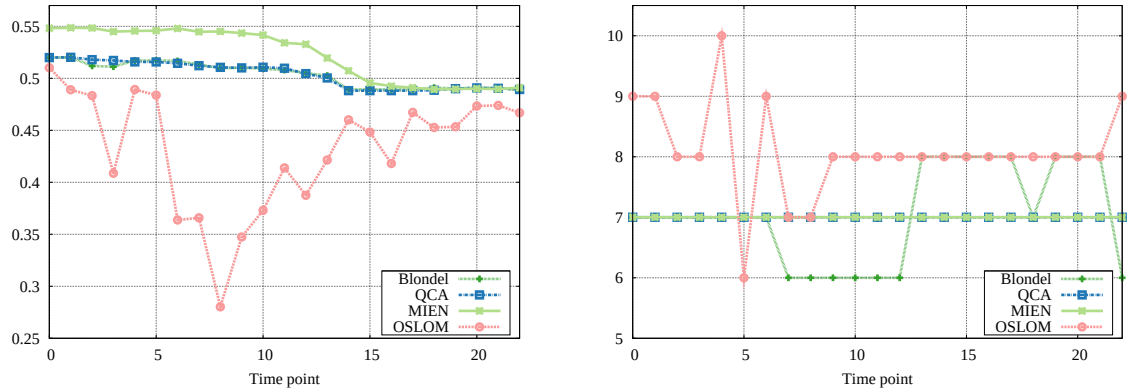
2.3.2 Results on real-world traces

We next present the results of QCA algorithms on real world dynamic social networks including ENRON email [98], arXiv e-print citation [22], and Facebook networks [100]. Due to the lack of appropriate communities corresponding to these traces, we report the performance of the aforementioned algorithms in reference to the static method proposed by Blondel et al. [6]. In particular, we will show the following quantities (1) modularity values, (2) the quality of the identified network communities through NMI scores, and (3) the processing time of our QCA in comparison with other methods. The above networks possess to contain strong community structures due to their high modularities, which was the main reason for them to be chosen.

For each network, time information is first extracted and a portion of the network data (usually the first snapshot) is then collected to form the basic network community structure. Our QCA method (aslo MIEN and OSLOM) take into account that basic community structure and run on the network changes whereas the static method has to be performed on the whole network snapshot for each time point. In this experiment, FacetNet method does not appear to complete the tasks in a timely manner, and is thus excluded from the plots.

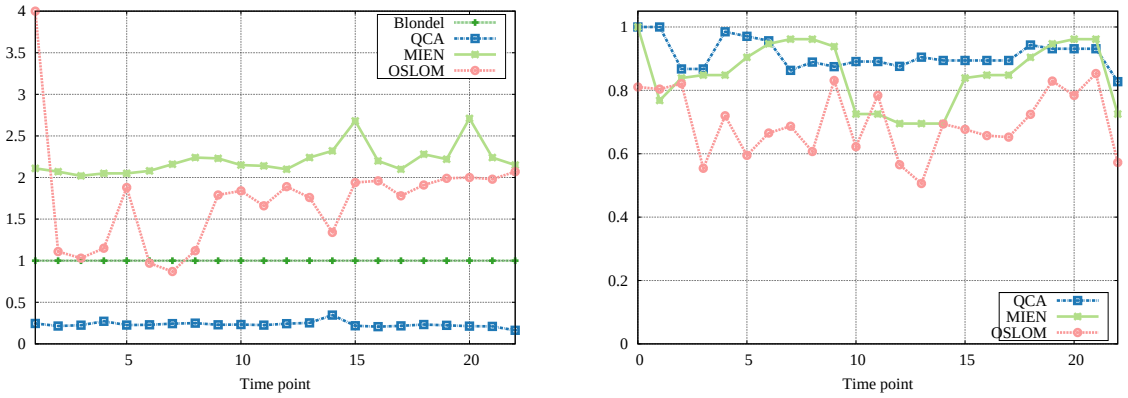
ENRON email network

Data. The Enron email network contains email messages data from about 150 users, mostly senior management of Enron Inc., from January 1999 to July 2002 [98]. Each email address is represented by an unique ID in the dataset and each link corresponds to a message between the sender and the receiver. After a data refinement process, we choose 50% of total links to form a basic community structure of the network with 7 major communities, and simulate the network evolution via a series of 21 growing snapshots.



A Modularity

B Number of Communities

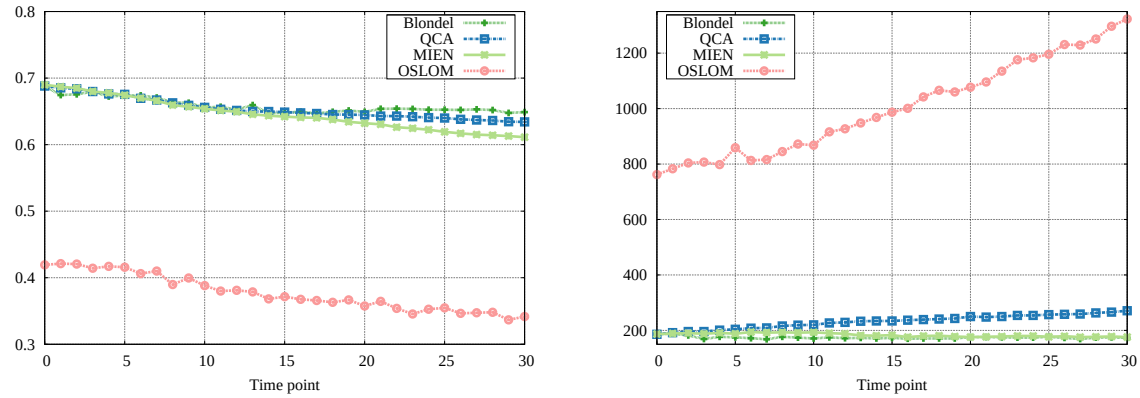


C Running Time(s)

D NMI

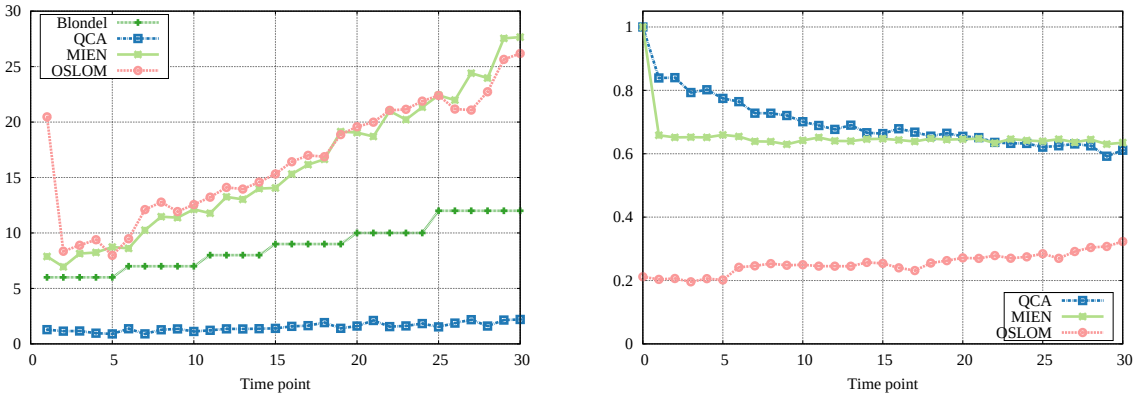
Figure 2-4. Simulation results on Enron email network.

Results. We first evaluate the modularity values computed by QCA, MIEN, OSLOM, and Blondel methods. As shown in Figure $2-4A$, our QCA algorithm archives competitively higher modularities than the static method but a little bit less than MIEN, and is far better than those obtained by OSLOM. Moreover, QCA also successes in maintaining the same numbers of communities of the other two methods MIEN and Blondel while OSLOM's are vague (Figure $2-4B$). In particular, the modularity values produced by QCA very well approximate those found by static method with lesser variation. There are reasons for that. Recall that our QCA algorithm takes into account the basic community structures detected by the static method (at the first snapshot) and processes on network changes only. Knowing the basic network community structure



A Modularity

B Number of Communities



C Running Time(s)

D NMI

Figure 2-5. Simulation results on arXiv e-print citation network.

is a great advantage of our QCA algorithm: it can avoid the hassle of searching and computing from scratch to update the network with changes. In fact, QCA uses the basic structure for finding and quickly updating the local optimal communities to adapt with changes introduced during the network evolution.

The running time of QCA and the static method in this small network are relatively close: the static method requires one second to complete each of its tasks while our QCA does not even ask for one (Figure 2 – 4*C*). In this dataset, MIEN and OSLOM requires a little more time (1.5 and 2.4 seconds in average for MIEN and OSLOM) to complete their tasks. Time and computational cost are significantly reduced in QCA

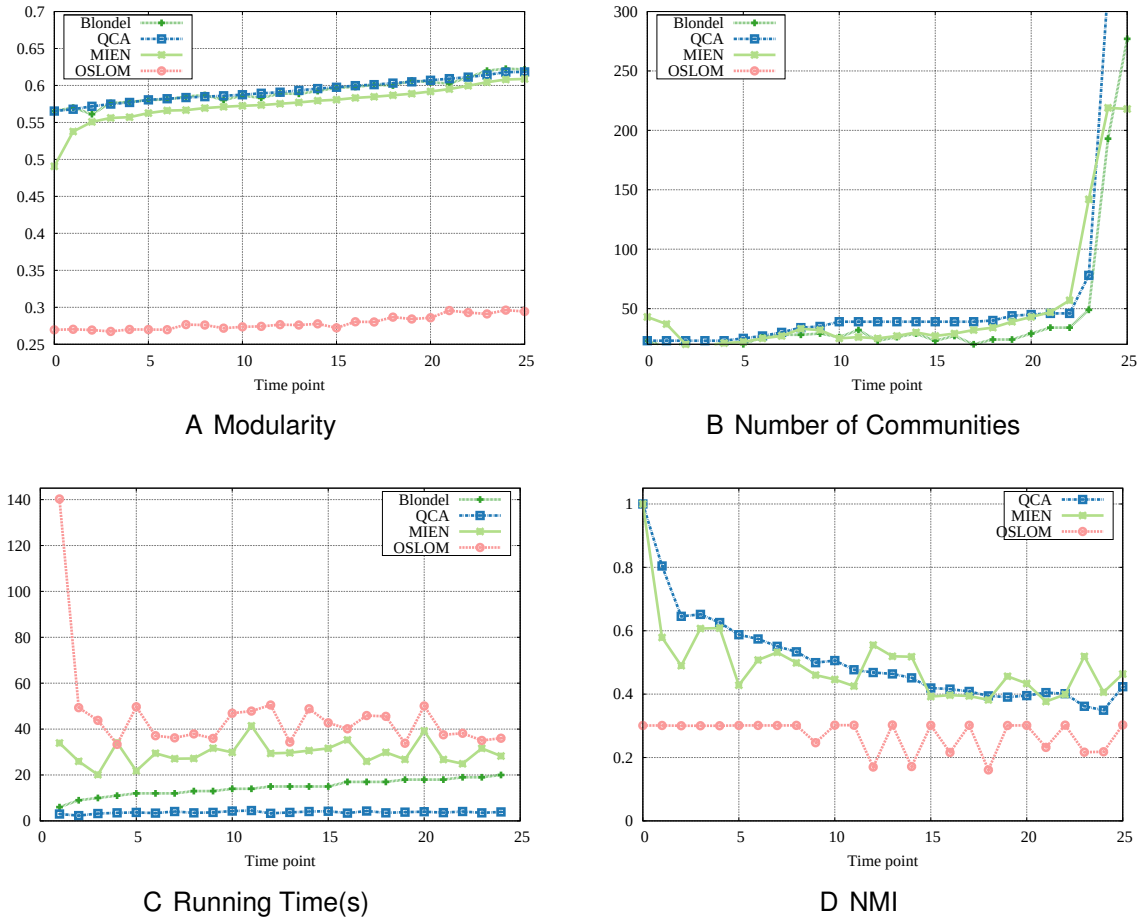


Figure 2-6. Simulation results on Facebook social network.

since our algorithms only take into account the network changes while the static method has to work on the whole network every time.

As reported in Figure $2-4D$, both the NMI scores of ours and MIEN method are very high and relatively close to 1 while those obtained by OSLOM fall short and are far from stable. These results indicate that in this Enron email network, both QCA and MIEN algorithms are able to identify high quality community structure with high modularity and similarity; however, only our method significantly reduces the processing time and computational requirement.

arXiv e-print citation network

Data. The arXiv e-print citation network [22] has become an essential mean of assessing research results in various areas including physics and computer sciences. This network contained more than 225K articles from January 1996 to May 2003. In our experiments, citation links of the first two years 1996 and 1997 were used to form the basic community structure of our QCA method. In order to simulate the network evolution, a total of 30 time dependent snapshots are created on a two-month regular basis from January 1998 to January 2003.

Results. We compare modularity results obtained by QCA algorithm at each network snapshot to Blondel as well as to MIEN and OSLOM methods. It reveals from Figure 2-5A that the modularities returned by QCA are very close to those obtained by the static method with much more stabler and are far higher than those obtained by OSLOM and MIEN. In particular, the modularity values produced by QCA algorithm cover from 94% up to 100% that of Blondel method and from 6% to 10% higher than MIEN and at least 1.5x better than OSLOM. In this citation networks, the numbers of communities detected by OSLOM take off with more than 1200 whereas those found by QCA, MIEN and Blondel methods are relatively small (Figure 2-5B). Our QCA method discovers more communities than both Blondel and MIEN as the network evolves and this can be explained based on the resolution limit of modularity [35]: the static method might disregard some small communities and tend to combine them in order to maximize the overall network modularity.

A second observation on the running time shows that QCA outperforms the static method as well as its competitor MIEN: QCA takes at most 2 seconds to complete updating the network structure while Blondel method requires more than triple that amount of time, MIEN and OSLOM asks for more than 5 times (Figure $2-5C$). In addition, higher NMI scores of QCA than MIEN's and especially OSLOM's scores (Figure $2-5D$) implies network communities identified by our approach are not only

of high similarity to the ground truth but also more precise than that detected by MIEN, while the computational cost and the running time are significantly reduced.

Facebook social network

Data. This dataset contains friendship information among New Orleans regional network on Facebook [100], spanning from September 2006 to January 2009 with more than 60K nodes (users) connected by more than 1.5 million friendship links. In our experiments, nodes and links from September 2006 to December 2006 are used to form the basic community structure of the network, and each network snapshot is recored after every month during January 2007 to January 2009 for a total of 25 network snapshots.

Results. The evaluation depicted in Figure $2-6A$ reveals that QCA algorithm achieves competitive modularities in comparison with the static method, and again far better than those obtained by MIEN and OSLOM method, especially in comparison with OSLOM whose perform was nice on synthesized networks. In the general trend, the line representing QCA results closely approximates that of the static method with much more stability. Moreover, the two final modularity values at the end of the experiment are relatively the same, which means that our adaptive method performs competitively with the static method running on the whole network.

Figure $2-6C$ describes the running time of the three methods on the Facebook data set. As one can see from this figure, QCA takes at least 3 seconds and at most 4.5 seconds to successfully compute and update every network snapshot whereas the static method, again, requires more than triple processing time. MIEN and OSLOM methods really suffer on this large scale network when requiring more than 10x and 11x that amounts of QCA running times. In conclusion, high NMI and modularity scores together with decent executing times on all test cases confirm the effectiveness of our adaptive method, especially when applied to real world social networks where a

centralized algorithm, or other dynamic algorithms, may not be able to detect a good network community structure in a timely manner.

However, there is a limitation of QCA algorithm we observe on this large network and want to point out here: As the the duration of network evolution lasts longer over time (i.e., the number of network snapshots increases), our method tends to divide the network into smaller communities to maximize the local modularity, thus results in an increasing number of communities and a decreasing of NMI scores. Figure $2-6B$ and $2-6D$ describes this observation. For instance, at snapshot 12 (a year after December 2006), the NMI score is approximately $1/2$ and continues decaying after this time point. It implies a refreshment of network community structure is required at this time, after a long enough duration. This is reasonable since activities on an online social network, especially on Facebook social network, tend to come and go rapidly and local adaptive procedures are not enough to reflect the whole network topology over a long period of time.

CHAPTER 3
OVERLAPPING COMMUNITY STRUCTURE DETECTION

In this chapter, we present AFOCS, an adaptive framework to discover and trace the evolution of network communities in dynamic complex systems. In section 3.1, we first state the problem definition including basic notations and the dynamic network model. Next, we present the procedure to detect the basic community structure in section 3.2, and then our AFOCS framework to update and trace the community structure evolution over time in section 3.3. Finally, we demonstrate the empirical results in section 3.4.

3.1 Problem Formulation

3.1.1 Basic notations

Let $G = (V, E)$ be an undirected unweighted graph representing the network where V is the set of N nodes and E is the set of M connections. Denote by $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ the network community structure, i.e., a collection of subsets of V where each $C_i \in \mathcal{C}$ and its induced subgraph form a community of G. In contrast with the disjoint community structure, we allow $C_i \cap C_j \neq \emptyset$ so that network communities can overlap with each other. For a node $u \in V$, let d_u, $N(u)$ and $Com(u)$ denote its degree, its neighbors and its set of community labels, respectively. For any $C \subseteq V$, let C^{in} and C^{out} denote the set of links having both endpoints in C and the set of links having exactly one endpoint in C, respectively. Finally, the terms node-vertex as well as edge-link-connection are used interchangeably.

3.1.2 Dynamic network model

Let $G_0 = (V_0, E_0)$ be the original input network and $G_t = (V_t, E_t)$ be a time dependent network snapshot recorded at time t. Denote by ΔV_t and ΔE_t the sets of nodes and edges to be added to or removed from the network at time t. Furthermore, let $\Delta G_t = (\Delta V_t, \Delta E_t)$ describe this change in terms of the whole network. The network snapshot at next time step $t + 1$ is expressed as a combination of the previous one

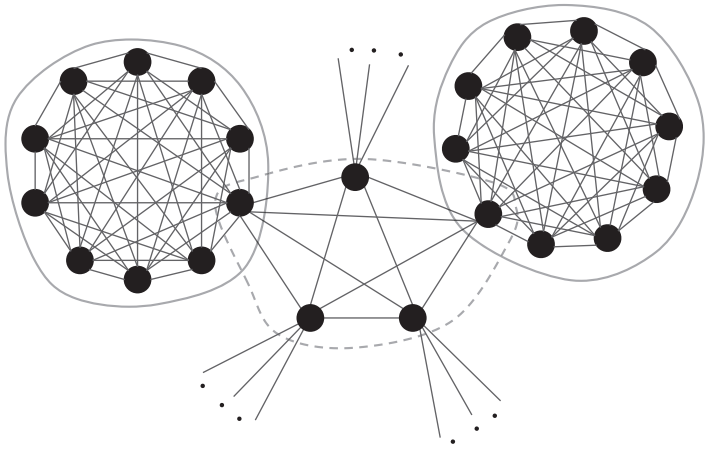


Figure 3-1. Overlapped v.s. non-overlapped community structures.

together with the change, i.e., $G_{t+1} = G_t \cup \Delta G_t$. Finally, a dynamic network $\mathcal{G}$ is defined as a sequence of network snapshots changing over time: $\mathcal{G} = (G_0, G_1, G_2, \ldots)$.

3.1.3 Density function

In order to quantify the goodness of an identified community, we use the popular density function Ψ [33] defined as: $\Psi(C) = \frac{2|C^{in}|}{|C|(|C|-1)}$ where $C \subseteq V$. Unlike the case of disjoint community structure, in which the number of connections crossing communities should be less than those inside them, our objective does not take into account the number of out-going links from each community. To understand the reason, let us consider a simple example pictured in Figure $3-1$. In the overlapping community structure point of view, it is clear that every clique should form a community on its own, and each community shares with the central clique exactly one node. However, in the disjoint community structure point of view, any vertex at the central clique has n internal and $2n$ external connections, which violates the concept of a community in the strong sense. Furthermore, the internal connectivity of the central clique is also dominated by its external density, which implies the concept of a community in weak sense is also violated. (A community C is in a weak sense if $|C^{in}| > |C^{out}|$, and in a strong sense if any node in C has more links inward than outward C [91]).

In order to set up a threshold on the internal density that suffices for a set of nodes C to be a local community, we propose a function $\tau(C)$ defined as follows:

$$\tau(C) = \frac{\sigma(C)}{\binom{|C|}{2}} \text{ where } \sigma(C) = \binom{|C|}{2}^{1-\frac{1}{\binom{|C|}{2}}}$$

Here $\sigma(C)$ is the threshold on the number of inner connections that suffices for C to be a local community. Particularly, a subgraph induced by C is a local community iff $\Psi(C) \geq \tau(C)$ or equivalently $|C^{in}| \geq \sigma(C)$. Several functions with the same purpose have been introduced in the literature, for instance, in the work of [56][62], and it is worth noting down the main differences between them and ours. First and foremost, our functions $\tau(C)$ and $\sigma(C)$ locally process on the candidate community C only and neither require any predefined thresholds or user-input parameters. Secondly, by Proposition 3.1, $\sigma(C)$ and $\tau(C)$ are increasing functions and closely approach C's full connectivity as well as its maximal density. That makes $\sigma(C)$ and $\tau(C)$ relaxation versions of the traditional density function, yet useful ones as we shall see in the experiments.

Proposition 3.1. *The function $f(n) = n^{1-\frac{1}{n}}$ is strictly increasing for $n \geq 4$ and $\lim_{n\to\infty} f(n) = n$.*

3.1.4 Objective function

Our objective is to find a community assignment for the set of nodes V which maximizes the overall internal density function $\Psi(\mathcal{C}) = \sum_{C\in\mathcal{C}} \Psi(C)$ since the higher the internal density of a community is, the clearer its structure would be. Although our objective puts more focus on the internal edges and less focus on the external edges, these external edges are not completely ignored but are considered in the following senses: they will be tested later for the formation of another community if the number of edges suffices. Only when these external edges are really sparse, they will not be considered.

3.1.5 Problem definition

Given a dynamic network $\mathcal{G} = (G_0, G_1, G_2, \ldots)$ where G_0 is the input network and $G_1, G_2, \ldots$ are network snapshots obtained through a collection of network topology changes $\Delta G_1, \Delta G_2, \ldots$ over time. The problem asks for an adaptive framework to efficiently detect and update the network overlapping community structure $\mathcal{C}_t$ at any time point t by only utilizing the information from the previous snapshot $\mathcal{C}_{t-1}$, as well as tracing the evolution of the network communities.

In the next section, we present our main contribution: an adaptive framework for (1) identifying basic overlapped community structure in a network snapshot and (2) updating as well as tracing the evolution of the network communities in a dynamic network model. First, we describe FOCS, a procedure to identify the basic communities in a static network, and then discuss in great detail how AFOCS adaptively updates these basic communities to cater with the evolution of the dynamic network.

3.2 Basic Community Structure Detection

We describe FOCS, the first phase of our framework that quickly discovers the basic overlapping network community structure. In general, FOCS works toward the classification of network nodes into different groups by first locating all possible densely connected parts of the network (3.2.1), and then combining those who highly overlap with each other, i.e., those share a significant substructure (3.2.2). Finally, a final refinement to group unassigned nodes into different communities is conducted in (3.2.3).

In FOCS, β (the input overlapping threshold) defines how much substructure two communities can share. Note that FOCS fundamentally differs from [1] in the way it allows $|C_i \cap C_j| \geq 2$ for any subsets C_i, C_j of V, and consequently allows network communities to overlap not only at a single vertex but also at a part of the whole community.

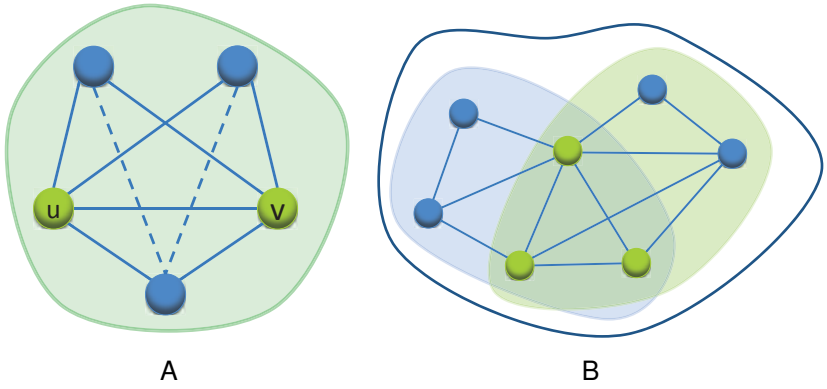


Figure 3-2. Locating and merging local communities.

3.2.1 Locating local communities

Local communities are connected parts of the network whose internal densities are greater than a certain level. In FOCS, this level is automatically determined based on the function $\tau()$ and the size of each corresponding part. Particularly, a local community is defined based on a connection (u, v) when the number of internal connections in the subgraph induced by $C \equiv \{u, v\} \cup (N(u) \cap N(v))$ exceeds $\sigma(C)$, or in other words, when $\Psi(C) \geq \tau(C)$ as illustrated in Figure 3-2A. Here, (a) A local community C defined by a link (u, v). Here $\Psi(C) = 0.9 > \tau(C) = 0.794$ (b) Merging two local communities sharing a significant substructure (OS score $= 1.027 > \beta = 0.8$).

However, there is a problem that might eventually arise: the containment of sub communities in an actual bigger one. Intuitively, one would like to detect a bigger community unified by smaller ones if the bigger community is itself densely connected. In order to filter this undesired case, we impose $\Psi\left(\bigcup_{i=1}^{s} C_i\right) < \tau\left(\bigcup_{i=1}^{s} C_i\right) \quad \forall s = 1 \ldots |\mathcal{C}|$ (note that some of these unifications do not contain all the nodes). In addition, we allow this locating procedure to skip over tiny communities of size less than 4. This condition is carried out from Proposition 3.1. This makes sense in terms of mobile or social networks where a group of mobile devices or a social community usually has size larger than 3, and intuitively agrees with the finding of [34][66]. Thus, the condition $|C| \geq 4$ is

imposed for any community C we discuss hereafter. The tiny communities will then be identified later. Alg. 6 describes this procedure.

Algorithm 6 Locating local communities

Input: $G = (V, E)$
Output: A collection of raw communities $\mathcal{C}_r$.

```
1: C_r ← ∅;
2: for ((u, v) ∈ E) do
3:     if (Com(u) ∩ Com(v) = ∅) then
4:         C ← {u, v} ∪ N(u) ∩ N(v);
5:         if (|C^in| ≥ σ(C) and |C| ≥ 4) then
6:             Check C's connectivity if |C| = 5;
7:             Define C a local community;
8:             /*Include C into the raw community structure*/
9:             C_r ← C_r ∪ {C};
10:        end if
11:    end if
12: end for
```

Lemma 7. *All local communities C's detected by Alg. 6 satisfy $\Psi(C) \geq \tau(4) \approx 0.74$. Furthermore, other communities satisfying these conditions will also be detected by Alg. 6.*

Proof. Alg. 6 will examine every edge $(u, v) \in E$ (except those whose endpoints are already in the same community), and by this greedy nature, any local community it detects has $|C| > 4$ and $\Psi(C) \geq \tau(C) \geq \tau(4) \approx 0.74$.

We now show that any community C statisfying $|C| \geq 4$ and $\Psi(C) \geq \tau(C) \geq \tau(4)$ will also be detected by Alg. 6. Suppose otherwise, that is there exists a community C satisfying these two conditions and is not detected by Alg. 6. To prove that this is not the case, we do the following: (1) Construct a community D which is not detected by Alg. 6 with $|D| = n \equiv |C|$ and $\Psi(D)$ is maximized, and (2) show that $\Psi(D) < \tau(D)$.

Because $|D| = |C|$, it implies $\tau(D) = \tau(C)$. However, since $\Psi(D)$ is maximized, $\Psi(D) \geq \Psi(C)$ which in turn implies $\Psi(C) \leq \Psi(D) < \tau(D) = \tau(C)$. This raises a contradiction to our original assumption, and thus concludes the proof.

To construct D, we do as follow (i) make D a clique of size n, and (ii) remove edges from D one by one until D cannot be detected by Alg. 6. By doing in this way, $\Psi(D)$ is maximized iff the number of removed edges is minimized.

It is easy to find the least number of edges we have to remove from D is $n/2$ if n is even and $n/2 - 1$ if n is odd. Therefore, $m_D = n(n-1)/2 - n/2$ if n is even, and $m_D = n(n-1)/2 - (n-1)/2$ if n is odd. Now, $\Psi(D) < \tau(D)$ iff $m_D < \left(\frac{n(n-1)}{2}\right)^{1-\frac{2}{n(n-1)}}$. Let $f(n)$ be the difference between the left and the right hand sides, we show that $f(n) < 0$ as n increases. Taking the derivative of $f(n)$ gives $\delta f(4) < 0$ and $f(n) < f(4) < 0$ for all even $n > 4$, and $\delta f(7) < 0$ and $f(n) < f(7) < 0$ for all odd $n > 7$. When $n = 5$, $f(5) > 0$ but this is the only exception and thus, can be handled easily in line 6 of Alg. 6. Therefore, we have $\Psi(D) < \tau(D)$, and hence, the conclusion follows. □

Theorem 3.1. *The local community structure $\mathcal{C}_r$ detected by Alg. 6 satisfies $\Psi(\mathcal{C}_r) \geq \tau(4) \times \Psi(OPT)$ where OPT is the optimal dense community assignment satisfying $\Psi(S) \geq \tau(4)$ for any $S \in OPT$.*

Proof. Let $\mathcal{C}_r$ be the local community structure returned by Alg. 6, and OPT be the optimal solution of the dense community assignment satisfying $\Psi(S) \geq \tau(4)$ for any $S \in OPT$. Let $k = |OPT|$. Clearly $\Psi(OPT) \leq k$. By Lemma 7, we know that Alg. 6 can detect as many communities as OPT but probably with less internal density. Moreover, since Alg. 6 only skips over edges in a community, it ensures that no real community is a substructure of a bigger one. Hence, we have $\Psi(\mathcal{C}_r) \geq \tau(4) \times k \approx 0.74 \times \Psi(OPT)$. This also implies that Alg. 6 is an 0.74-approximation algorithm for finding local densely connected communities. □

Lemma 8. *The time complexity of Alg. 6 is $O(dM)$ where $d = \max_{v \in V} d_v$.*

Proof. Time to examine an edge (u, v) is $|N(u)| + |N(v)| = d_u + d_v$. However, when u and v are in the same community, (u, v) will be skipped. Therefore, the total time complexity is upper bounded by $d \sum_{u \in V} d_u = O(dM)$. □

3.2.2 Combining overlapping communities

After Alg. 6 finishes, the raw network community structure is pictured as a collection of (possibly overlapped) dense parts of the network together with outliers. As some of those dense parts can possibly share significant substructures, we need to merge them if they are highly overlapped. To this end, we introduce the overlapping score of two communities defined as follow

$$OS(C_i, C_j) = \frac{|I_{ij}|}{\min\{|C_i|, |C_j|\}} + \frac{|I_{ij}^{in}|}{\min\{|C_i^{in}|, |C_j^{in}|\}}$$

where $I_{ij} = C_i \cap C_j$. Basically, $OS(C_i, C_j)$ values how important the common nodes and links shared between C_i and C_j mean to the smaller community. In comparison with the distance metric suggested in [63], our overlapping score not only takes into account the fraction of common nodes but also values the fraction of common connections, which is crucial in order to combine network communities. Furthermore, $OS(\cdot, \cdot)$ is symmetric and scales well with the size of any community, and the higher the overlapping score, the more those communities in consideration should be merged. In this merging process, we combine communities C_i and C_j if $OS(C_i, C_j) \geq \beta$ (Figure 3-2B).

Algorithm 7 Combining local communities

Input: Raw community structure $\mathcal{C}_r$
Output: A refined community structure $\mathcal{D}$.

```
1: D ← C_r;
2: Done ← false;
3: while (!Done) do
4:     Done ← true;
5:     Order (C_i, C_j)'s by their OS(C_i, C_j) scores;
6:     for (C_i, C_j ∈ C_r) do
7:         if (OS(C_i, C_j) > β and ) then
8:             C ← Combine C_i and C_j;
9:             /*Update the current structure*/
10:            D ← (C_f \ {C_i ∪ C_j}) ∪ C;
11:            Done ← False;
12:        end if
13:    end for
14: end while
```

The time complexity of Alg. 7 is $O(N_0^2)$ where N_0 is the number of local communities. Clearly, $N_0 \leq M$ and thus, it can be $O(M^2)$. However, when the intersection of two communities is upper bounded, by Lemma 9 we know that the number of local communities is also upper bounded by $O(N)$, and thus, the time complexity of Alg. 7 is $O(N^2)$. In our experiments, we observe that the running time of this procedure is, indeed, much less than $O(N^2)$.

Lemma 9. *The number of raw communities detected in Alg. 6 is $O(N)$ when the number of nodes in the intersection of any two communities is upper bounded by a constant α.*

Proof. For each $C_i \in \mathcal{C}$, decompose it into overlapped and non-overlapped parts, denoted by C_i^{ov} and C_i^{nov}. We have $C_i = C_i^{ov} \cup C_i^{nov}$ and $C_i^{ov} \cap C_i^{nov} = \emptyset$. Therefore, $|C_i| = |C_i^{ov}| + |C_i^{nov}|$.

Now,

$$\sum_{C_i \in \mathcal{C}} |C_i| = \sum_{C_i \in \mathcal{C}} (|C_i^{ov}| + |C_i^{nov}|) \leq N + \sum_{i<j} |C_i^{ov} \cap C_j^{nov}|,$$

where $N = \sum_{C_i \in \mathcal{C}} |C_i^{nov}| + \left|\bigcup_{C_i \in \mathcal{C}} |C_i^{ov}|\right|$. For an upper bound of the second term, rewrite

$$\sum_{i<j} |C_i^{ov} \cap C_j^{nov}| \leq N + \sum_{|C_i \cap C_j| \geq 2} |C_i \cap C_j| \leq N(1+\alpha),$$

where $\alpha = \max\{|C_i \cap C_j| : |C_i \cap C_j| \geq 2\}$

Hence, $\sum_{C_i \in \mathcal{C}} |C_i| \leq N(2+\alpha)$. Let N_0 be the number of raw communities, it follows that $N_0 \min\{|C_i|\} \leq \sum_{C_i \in \mathcal{C}} |C_i| \leq (2+\alpha)N$. Since $\min\{|C_i|\} \geq 4$, we have $N_0 \leq \frac{(2+\alpha)}{4} N = O(N)$. □

Remark

After the above community merging process, detected communities can possibly be of very large sizes. The explanation is as follow: small quasi-cliques are discovered in the first phase (Alg. 6) as densely connected parts of the network, and are regarded as candidate elements for bigger communities in the merging process. If these small cliques are loosely connected to the rest of the network, they will retain as local

communities afterwards. Otherwise, they can be merged to other dense parts to become new bigger communities. As a result, if the communities are highly overlapped, some of them can potentially grow to very large sizes at the end of the merging process, beside the small cliques detected at the first place. Larger dense quasi-cliques, though rare in many networks, will surely be detected by FOCS as we observed in Theorem 3.1.

3.2.3 Revisiting unassigned nodes

Even when the above two procedures are executed, there would still exist leftover nodes or edges due to their less attraction to the rest of the network. Because of its size constraint, the first procedure skips over tiny communities of sizes less than four and thus, may leave out some nodes unlabeled. These nodes will not be touched in the second phase since they do not belong to any local communities, and consequently, will remain unassigned afterwards. Moreover, they are mostly nodes with less connection to the rest of the network, and thus, are very likely supplement nodes possibly to their adjacent communities. Therefore, we need to revisit those nodes to either group them into appropriate communities or classify them as outliers based on their connectivity structures.

Algorithm 8 Revisit Unassigned Nodes

Input: The refined community structure $\mathcal{D} = \{D_1, D_2, \ldots, D_t\}$
Output: The basic community structure $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$

1: $\mathcal{C} \leftarrow \mathcal{D}$;
2: **for** $(u \in V$ and $Com(u) == \emptyset)$ **do**
3: $\quad NC(u) \leftarrow \{C_j \in \mathcal{C} | u$ is adjacent to $C_j\}$;
4: $\quad$ **for** $(C_j \in NC(u))$ **do**
5: $\quad\quad$ **if** $(F_{C_j \cup \{u\}} \geq F_{C_j})$ **then**
6: $\quad\quad\quad C_j \leftarrow C_j \cup \{u\}$;
7: $\quad\quad\quad Com(u) \leftarrow Com(u) \cup \{j\}$;
8: $\quad\quad$ **end if**
9: $\quad$ **end for**
10: $\quad$ **if** $(Com(u) == \emptyset)$ **then**
11: $\quad\quad$ Classify u as an outlier;
12: $\quad$ **end if**
13: **end for**

Alternatively, this process can be thought of as a community trying to hire adjacent unassigned nodes which are similar to the host community. However, the internal density function might be too strict for them to be included in any community (which was also the reason why they are left unassigned). To this end, we need a community fitness function in order to quantify the similarity between a node u and a neighbor community C. We find the fitness function $F_S = \frac{|S^{in}|}{2|S^{in}|+|S^{out}|}$ (where $S \subseteq V$) commonly used in [56][39][63] performs competitively in both synthesized and real-world datasets. Taking into account this fitness function, a community C will keep hiring any unassigned adjacent vertex of maximum similarity in a greedy manner, provided the newly joined vertex does not shrink down the community's current fitness value. If there is no such node, C is defined as a final network community. Nodes remained unlabeled through this last procedure are identified as outliers. This algorithm is presented in Alg. 8.

3.3 Detecting Evolving Network Communities

We describe AFOCS, the second phase and also the main focus of our detection framework. In particular, we use AFOCS to adaptively update and trace the network communities, which were previously initialized by FOCS, as the dynamic network evolves over time. Note that FOCS is executed only once on G_0, after that AFOCS will take over and handle all changes introduced to the network.

Let us first discuss the various behaviors of the community structure when the network topology evolves over time. Suppose $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, .., C_n\}$ is the current network and its corresponding overlapping community structure, respectively. We use the term intra links to denote edges whose two endpoints belong to the same community, inter links to denote those with endpoints connecting different disjoint communities and the term hybrid links to stand for the others. For each community C of G, the number of connections joining C with the others are lesser than the number of connections within C itself by definition

Intuitively, the addition of intra links or removal of inter links between communities of G will strengthen them and consequently, will make the structure of G more clear. Similarly, removing intra links from or introducing inter links to a community of G will decrease its internal density and as a result, loosen its internal structure. However, when two communities have less distraction to each other, adding or removing links makes them more attractive to each other and therefore, leaves a possibility that they can overlap with each other or can be combined to form a new community. The updating process, as a result, is very complicated and challenging since any insignificant change in the network topology could possibly lead to an unpredictable transformation of the network community structure.

In order to reflect these changes to a complex network, its underlying graph model is frequently updated by either inserting or removing a node or a set of nodes, or an edge or a set of edges. A scrutiny look into these events reveals that the introduction or removal of a set of nodes (or edges) can furthermore be decomposed as a collection of node (or edge) insertions (or removals), in which only a node (or only an edge) is inserted (or removed) at a time. Therefore, changes to the network at each time step can be viewed as a collection of simpler events whose details are as follow:

- newNode $(V + u)$: A new node u and its adjacent edge(s) are introduced
- removeNode $(V - u)$: A node u and its adjacent edge(s) are removed from the network.
- newEdge $(E + e)$: A new edge e connecting two existing nodes is introduced.
- removeEdge $(E - e)$: An edge e in the network is removed.

As we mentioned earlier, our adaptive framework initially requires a basic community structure $\mathcal{C}_0$. To obtain this basic structure, we apply FOCS algorithm at the first network snapshot, i.e., we execute FOCS on the network G_0 and then let AFOCS adaptively handle this structure as the network evolves.

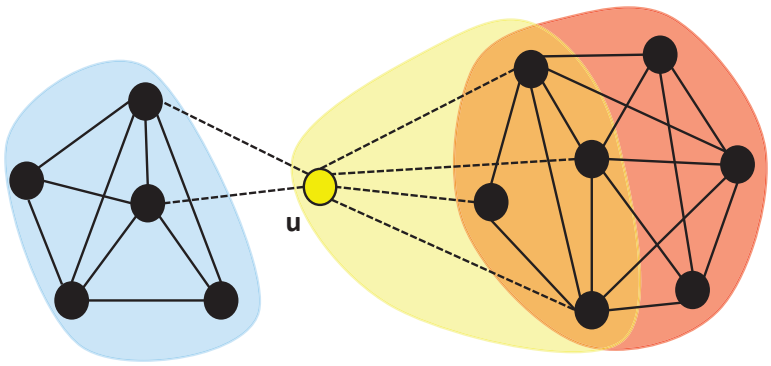


Figure 3-3. A possible scenario when a new node is introduced.

3.3.1 Handling a new node

Let us discuss the first case when a new node u and its associated links are introduced to the network. Possibilities are (1) u may come with no adjacent edge or (2) with many of them connecting one or more possibly overlapped communities. If u has no adjacent edge, we simply join u in the set of outliers and preserve the current community structure.

The interesting case happens, and it usually does, when u comes with multiple links connecting one ore more existing communities. Since network communities can overlap each other, we need to determine which ones u should join in in order to maximize the gained internal density. But how can we quickly and effectively do so? By Lemma 10, we give a necessary condition for a new node in order to join in an existing community, i.e., our algorithm will join node u in C if the number of connections u has to C suffices: $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$. However, failing to satisfy this condition does not necessarily imply that u should not belong to C, since it can potentially gather some substructure of C to form a new community (Figure 3-3). Thus, we also need to handle this possibility. Alg. 9 presents the algorithm.

Lemma 10. *Suppose u is a newly introduced node with d_{ui} connections to each adjacent community C_i. u will join in C_i if $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$.*

Algorithm 9 Handling a new node u

Input: The current community structure $\mathcal{C}_{t-1}$
Output: An updated structure $\mathcal{C}_t$.

$C_1, C_2, \ldots, C_k \leftarrow$ Adjacent communities of u;
for i = 1 **do** to k
  if $(d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\})$ **then**
    $C_i \leftarrow C_i \cup \{u\}$;
  else
    $C \leftarrow N(u) \cap C_i$;
    if $(\Psi(C) \geq \tau(C)$ and $|C| \geq 4)$ **then**
      $C_i \leftarrow C_i \cup \{u\}$;
    end if
  end if
end for
/*Checking new communities formed from outliers*/
for $(v \in N(u)$ and $Com(v) \cap Com(u) = \emptyset)$ **do**
  $C \equiv N(u) \cap N(v)$;
  if $(\Psi(C) \geq \tau(C)$ and $|C| \geq 4)$ **then**
    Define C a new community;
  end if
end for
Merging overlapping communities on $C_1, C_2, \ldots, C_k$;
Update $\mathcal{C}_t$;

Proof. Prior to u joining to C_i, the internal density is $\Psi(C_i) = \frac{2|C_i^{in}|}{|C_i|(|C_i|-1)}$. Similarly, after u joining in C_i, the density function is $\Psi(C_i \cup \{u\}) = \frac{2|C_i^{in}|+2d_{ui}}{|C_i|(|C_i|+1)}$. Taking the difference between these two quantities gives $\Psi(C_i \cup \{u\}) > \Psi(C_i) \iff d_{ui} > \frac{2|C_i^{in}|}{|C_i|-1}$. Moreover, u should also satisfy $\Psi(C_i \cup \{u\}) \geq \tau(C_i \cup \{u\})$, which in turn implies $d_{u,i} \geq f(|C_i|+1) - |C_i^{in}|$. Therefore, $d_{ui} > \max\{\frac{2|C_i^{in}|}{|C_i|-1}, f(|C_i|+1) - |C_i^{in}|\}$. □

The analysis of Alg. 9 is shown by Lemma 11. In particular, we show that this procedure achieves at least 74% the internal density of the optimal assignment for u, given the prior community structure.

Lemma 11. *Alg. 9 produces a community assignment that, prior to the community combination process, achieves* $\Psi(\mathcal{C}_t) \geq \tau(4) \times \Psi(OPT(u)_t)$ *where* $OPT(u)_t$ *is the optimal community assignment for* u *at time* t*, given the prior community structure* $\mathcal{C}_{t-1}$.

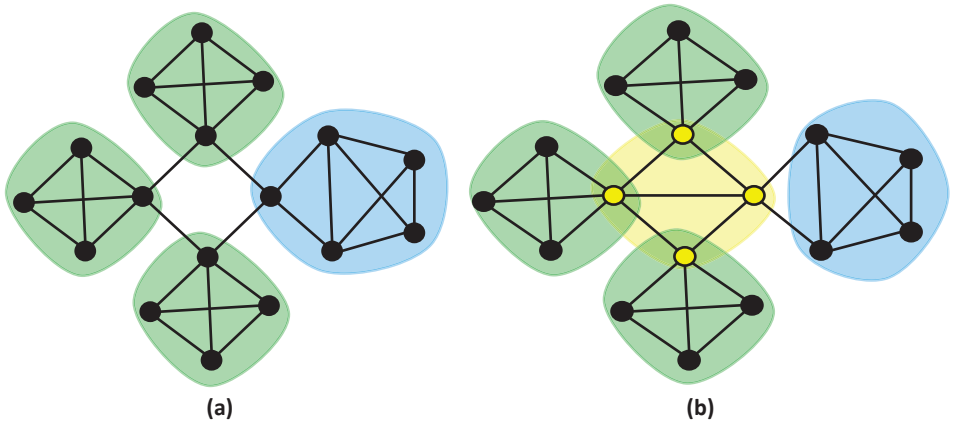


Figure 3-4. Possible scenarios when a new edge is introduced.

Proof. Let $C_1, C_2, \ldots, C_k$ be the communities (including the newly formed ones) in $\mathcal{C}_t$ that Alg. 9 assigns the new node u to. Note that in the optimal solution $OPT(u)_t$, the number of communities u belongs to should not exceed k since each C_i is also a candidate for $OPT(u)_t$ (of course, $OPT(u)_t$ could possibly rearrange nodes differently). Therefore, the optimal internal density gained is upper bounded by k. On the other hand, Alg. 9 makes sure that each community C_i that u joins in should have $\Psi(C_i) \geq \tau(C_i) \geq \tau(4)$ since $|C_i| \geq 4$. Thus, Alg. 9 will achieve at least $\tau(4) \times k \approx 0.74 \times \Psi(OPT(u)_t)$. □

3.3.2 Handling a new edge

In case where a new edge $e = (u, v)$ connecting two existing vertices u and v is introduced, we divide it further into two four smaller cases: (1) e is solely inside a single community C (2) e is within the intersection of two (or more) communities (3) e is joining two separated communities and (4) e is crossing overlapped communities. If e is totally inside a community C, its presence will strengthen C's internal density and by Lemma 12, we know that adding e should not split the current community C into smaller substructures.

In the second subcase, the introduction of the new edge might increase the density of some part of C and it is reasonable to think of that part (say D) as a new separated community. However, since D originally shared a significant substructure with C, the

merging process will then combine C and D (if they were separated) to be a bigger community, thus raising the same community as if C was kept intact. Therefore, the same reaction applies in the second subcase when e is within the intersection of two communities since their inner densities are both increased. Thus, in these first two cases, we leave the current network structure intact.

Algorithm 10 Handling a new edge (u, v)

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if ($(u, v) \in$ a single community OR $(u, v) \in C_u \cap C_v$) **then**
    $\mathcal{C}_t \leftarrow \mathcal{C}_{t-1}$;
else if $(Com(u) \cap Com(v) == \emptyset)$ **then**
    $C \leftarrow N(u) \cap N(v)$;
    if $(\Psi(C) \geq \tau(C))$ **then**
        Define C a new community;
        Check for combining on $Com(u)$, $Com(v)$ and C;
    else
        for $(D \in Com(u)$ (or $D* \in Com(v)))$ **do**
            if $(\Psi(D \cup \{v\}) \geq \tau(D \cup \{v\}))$ (or $\Psi(D* \cup \{u\}) \geq \tau(D* \cup \{v\}))$ **then**
                $D \leftarrow D \cup \{v\}$ (or $D* \leftarrow D* \cup \{u\}$)
            end if
        end for
        Merging overlapping communities for D's (or $D*$);
    end if
    Update $\mathcal{C}_t$;
end if

Handling the last two subcases is complicated since any of them can either have no effect on the current network structure or unpredictably form a new network community, and furthermore can overlap or merge with the others (Figure 3-4). However, there is still a possibility that the introduction of this new link, together with some substructure of C_u or C_v, suffices to form a new community that can overlap with not only C_u and C_v but also with some of the others. The other subcases can be handled similarly. Alg. 10 describe this procedure.

Lemma 12. *If an new edge (u, v) is introduced solely inside a community C, it should not split C into smaller substructures.*

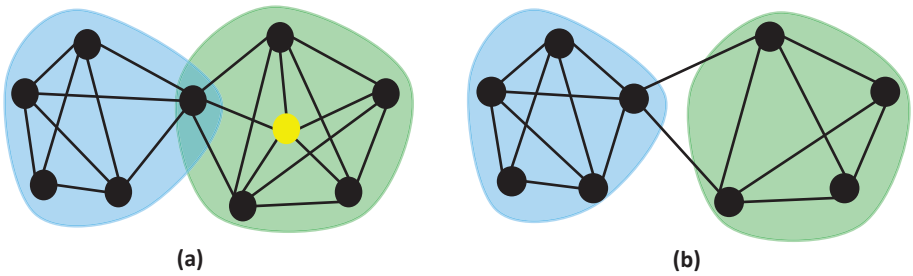


Figure 3-5. Possible scenarios when an existing node is removed.

Proof. Suppose otherwise, that is C is divided into smaller parts C_1 and C_2. Prior to the introduction of (u, v), we have $\Psi(C) = \Psi(C_1 \cup C_2) \geq \tau(C) = \tau(C_1 \cup C_2)$. Now, when C_1 and C_2 are formed, they imply that $\Psi(C_1 \cup C_2 + (u, v)) < \tau(C_1 \cup C_2 + (u, v))$. Putting all together, we have $\tau(C_1 \cup C_2 + (u, v)) = \tau(C_1 \cup C_2) > \Psi(C_1 \cup C_2 + (u, v)) > \Psi(C) > \tau(C_1 \cup C_2)$, which raises a contradiction. Thus, the conclusion follows. □

3.3.3 Removing an existing node

When an existing node u is about to be removed from the network, all of its adjacent edges will also be removed as a consequence. If u is an outlier, we can simply exclude u and its corresponding links from the current structure and safely keep the network communities unchanged.

In unfortunate situations where u is not an outlier, the problem becomes very challenging in the sense that the resulting community is complicated: it can either be unchanged, or broken into smaller communities, or could probably be further merged with the other communities. To give a sense of this effect, let's consider two examples illustrated in Figure 3-5. In the first example, when C is almost a full clique, the removal of any node will not break it apart. However, if we a remove node that tends to connect the others within a community, the leftover module is broken into a smaller one together with a node that will later be merged to one of its nearby communities. Therefore, identifying the leftover structure of C is a crucial task once a vertex u in C is removed.

To quickly handle this task, we first examine the internal density of C excluding the removed node u. If the number of internal connections still suffices, e.g., $\Psi(C\backslash\{u\}) \geq \tau(C\backslash\{u\})$, we can safely keep the current network community structure intact because C is still tightly connected itself with a sufficient internal density. Otherwise, this community is of a weak strength and shall be broken into smaller ones. These substructures might further be merged with other communities if C origianlly overlaps with them. To efficiently detect these new substructures, we apply Alg. 6 on the subgraph induced by $C\backslash\{u\}$ to quickly identify the leftover modules in C, and then let these modules hire a set of unassigned nodes $\Psi(C)$ that help them increasing their inner densities. Finally, we locally check for community combination, if any, by using an algorithm similar to Alg. 7. Alg. 11 presents the procedure.

Algorithm 11 Removing a node u

Input: The current community structure $\mathcal{C}_{t-1}$.
Output: An updated structure $\mathcal{C}_t$.

```
for (C ∈ Com(u) and Ψ(C\{u}) < τ(C\{u})) do
    LC ← Local communities by Alg 6 on C\{u};
    for (C_i ∈ LC and |C_i| ≥ 4) do
        S_i ← Nodes such that Ψ(C_i ∪ S_i) ≥ τ(C_i ∪ S_i);
        C_i ← C_i ∪ S_i;
    end for
    Merging overlapping communities on LC;
end for
Update C_t;
```

3.3.4 Removing an edge

In the last situation when an edge $e = (u, v)$ is about to be removed, we divide it further into four subcases similar to those of a new edge (1) e is between two disjoint communities (2) e is inside a sole community (3) e is within the intersection of two (or more) communities and finally (4) e is crossing overlapping communities.

In the first subcase, when e is crossing two disjoint communities, its removal will make the network structure more clear (since we now have less connections between groups), and thus, the current communities should be keep unchanged. When e is

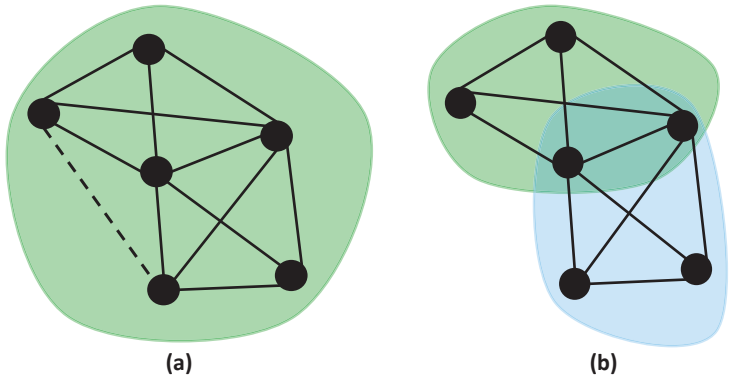


Figure 3-6. Possible scenarios when an existing edge is removed.

totally within a sole community C, handling its removal is complicated since this can lead to an unpredictable transformation of the host module: C could either be unchanged or broken into smaller modules if it contains substructures which are less attractive to each other, as depicted in Figure 3-6. Therefore, the problem of identify the structure of the remaining module becomes the central part for not only this case but also for the others.

Algorithm 12 Removing an edge (u, v)

Input: The current structure $\mathcal{C}_{t-1}$.
Output: An updated community structure $\mathcal{C}_t$.

if $((u, v)$ is an isolated edge) **then**
    $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash \{u, v\}) \cup \{u\} \cup \{v\}$;
else if $(d_u = 1$ (or $d_v = 1))$ **then**
    $\mathcal{C}_t = (\mathcal{C}_{t-1} \backslash C(u)) \cup \{u\} \cup C(v)$;
else if $(C \equiv C(u) \cap C(v) = \emptyset)$ **then**
    $\mathcal{C}_t = \mathcal{C}_{t-1}$;
else if $(\Psi(C \backslash (u, v)) < \tau(C \backslash (u, v)))$ **then** /*Here $C \neq \emptyset$*/
    $LC \leftarrow$ Local communities by Alg 6 on $C \backslash (u, v)$;
    Define each $L \in LC$ a local community of $\mathcal{C}_{t-1}$;
    Merging overlapping community on L's;
end if
Update $\mathcal{C}_t$;

To quickly handle these tasks, we first verify the inner density of the remaining module and, again utilize the local community location method (Alg. 6) to locally identify the leftover substructures. Next, we check for community combination since

these structures can possibly overlap with existing network communities. The detailed procedure is described in Alg. 12.

3.3.5 Remarks

Note that the ultimate goal of our framework is to adaptively detect and update the community structure as the network evolves, i.e., to mainly deal with the dynamics of a mobile network. As a result, we mainly put our focus on AFOCS. Although FOCS, the first detection phase, appears to be a centralized algorithm, it is executed only once at the very first network snapshot whereas AFOCS stays up and locally handles all changes as the network evolves over time. That said, we do not execute FOCS again. Furthermore, AFOCS can be run independently with FOCS, i.e., one can use any localized detection algorithm to identify a basic community structure at the first phase. Thus, AFOCS can be easily apply to solve mobile network problems.

3.3.6 Complexity

Our main algorithm consists of two parts: (1) finding the basic community structure and (2) updating the network community structure through changes introduced at every time step. The complexity of quickly unfolding the basic network community structures has been claimed to be linear in terms of number of nodes and links $O(M + N)$ [58]. To handle the case of a new node of degree p coming in, our algorithm computes p forces this new node applies to its neighbors, which results in linear time complexity $O(p)$. When a new edge connecting nodes u and v is introduced to the network, our algorithm just simply computes the forces applied to communities adjacency nodes, which takes $O(|C(u)| + |C(v)|)$ in the best case and $O(k \times \max\{|C(u)|, |C(v)|\})$ in the worst case when some nodes in a module are pulled out to form new communities (where k is the number of communities in G). The time taken to handle the last two cases is essentially the time complexity of the clique percolation, which is roughly $O(|C(u)|^3)$ in the worst case. Although the time complexity is in the third order of number of nodes, the total nodes inside a single community is relatively small in comparison with the total number

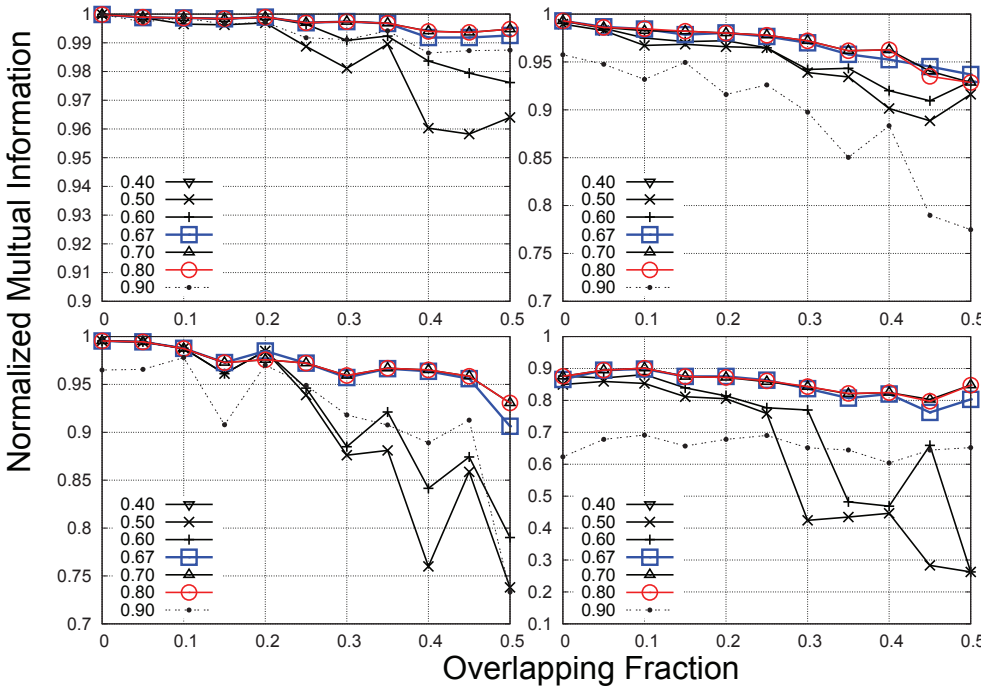


Figure 3-7. NMI scores for different values of β. $N = 5000$ (top), $N = 1000$ (bottom), $\mu = 0.1$ (left), $\mu = 0.3$ (right).

of vertices N, and thus, does not affect the actual running time. Experimental results in Section 4 show that our algorithm performs quickly and smoothly in large social online networks.

3.4 Experimental Results

In this section, we first present the empirical results of AFOCS in comaprison with two static detection methods: CFinder - the most popular method [84], and COPRA - the most effective method [40]. We next compare the performance of AFOCS with other dynamic methods including OSLOM [60], FacetNet [71] and iLCD [9].

Data Sets: We use networks generated by the well-known LFR overlapping benchmark [58], the 'de facto' standard for evaluating overlapping community detection algorithms. Generated networks follow power-law degree distributions and contain embedded overlapping communities (the ground truth) of varying sizes that capture the internal characteristics of real-world networks.

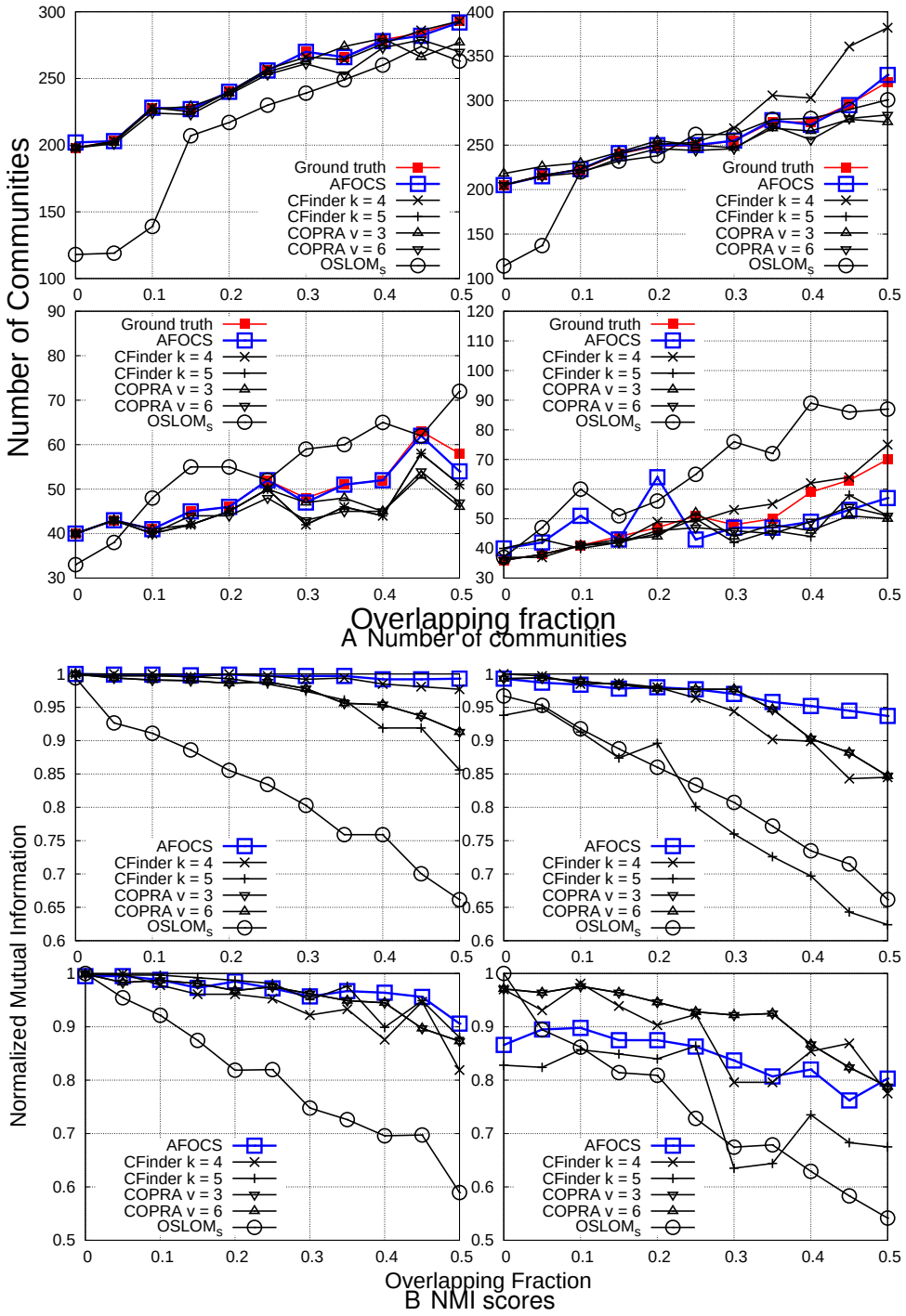


Figure 3-8. Comparison among AFOCS, COPRA and CFinder methods. $N = 5000$ (top), $N = 1000$ (bottom), $\mu = 0.1$ (left), $\mu = 0.3$ (right).

Set up: To fairly compare with COPRA and to avoid being biased, we keep the parameters close to [40]: the minimum and maximum community sizes are $c_{min} = 10$ and $c_{max} = 50$, each vertex belongs to at most two communities, $o_m = 2$. $N = 1000$ and $N = 5000$. The mixing rate $\mu = 0.1$ and $\mu = 0.3$. The overlapping fraction γ, which determines the fraction of overlapped nodes, is from 0 to 0.5. Since COPRA is nondeterministic, we run it 10 times on each instance and select the best result.

Metrics: We evaluate following metrics.

(1) The generalized Normalized Mutual Information (NMI) [56] specially built for overlapping communities. NMI scores the similarity between the detected network communities and the ground truth. This is an standardized measure since NMI(U,V)=1 if structures U and V are identical and 0 if they are totally separated.

(2) The number of communities, ignoring singleton communities and unassigned nodes. A good community detection method should produce roughly the same number of communities with the known ground truth.

3.4.1 Choosing the overlapping threshold β

The overlapping threshold β is the only input parameter required by our framework, and thus, determining its appropriate value plays an important role in assessing AFOCS's performance. To best determine this threshold, we run AFOCS on generated networks with different values of β, and record the similarities between the detected communities and the ground-truth via NMI scores (Figure 3-7). Of course, the higher NMI scores imply the better β values.

As a threshold parameter, β controls how much substructure communities can have in common. The smaller values of β imply the more we allow network communities to overlap with each other, and vice versa. Similarly, β can be thought of as the zooming scale of the network structure where lower β's reveal the coarser and higher β's reveal the finer structure. As depicted in Figure 3-7, the best values for β are ranging from 0.67 to 0.80, among which $\beta = 0.70$ yields the best community similarity (NMI scores are

ranging from 0.8 to 1) in all of the generated networks. Therefore, we fix the overlapping threshold in AFOCS to be 0.70 hereafter.

3.4.2 Reference to static methods

We show our results in groups of four. For each case we vary the overlapping fraction γ from 0 to 0.5 and analyze the results found by AFOCS, CFinder, COPRA and (static) OSLOM methods (OSLOM$_s$). We only present results when corresponding parameters give top performance for CFinder (clique size $k = 4, 5$) and COPRA (max. communities per vertex $v = 3, 6$).

Figure 3-8A shows the number of communities found by AFOCS, COPRA and CFinder, OSLOM$_s$ and the ground truth. It reveals from this figure that the numbers of communities found by AFOCS, marked with squares, are the closest and almost identical to the ground truth as the overlapping fraction gets higher. There is an exception when $N = 1000$ and $\mu = 0.3$ which we will discuss later. In terms of NMI scores, as one can infer from Figure 3-8B, AFOCS achieves the highest performance among all methods with much more stable. A common trend in this test is the performances of all methods degrade (1) when the mixing rate μ increases, i.e., when the community structure becomes more ambiguous or (2) when the size of network decreases while the mixing rate μ stays the same. Even though AFOCS is not very competitive only when both negative factors happen in the bottom-right char as $N = 1000$ and $\mu = 0.3$, it is in general the best performer. OSLOM$_s$, the static version of OSLOM method, does not appear to perform well on these synthesized data as its NMI scores are low and degrade quickly when the network communities become more stochastic. The NMI scores of AFOCS, on the other hand, remain high and stable even when the network community structure becomes unclear when the overlapping fraction increases.

The significant gap is observed when the mixing rate gets higher ($\mu = 0.3$) and the network size gets smaller ($N = 1000$). AFOCS provides less numbers of communities

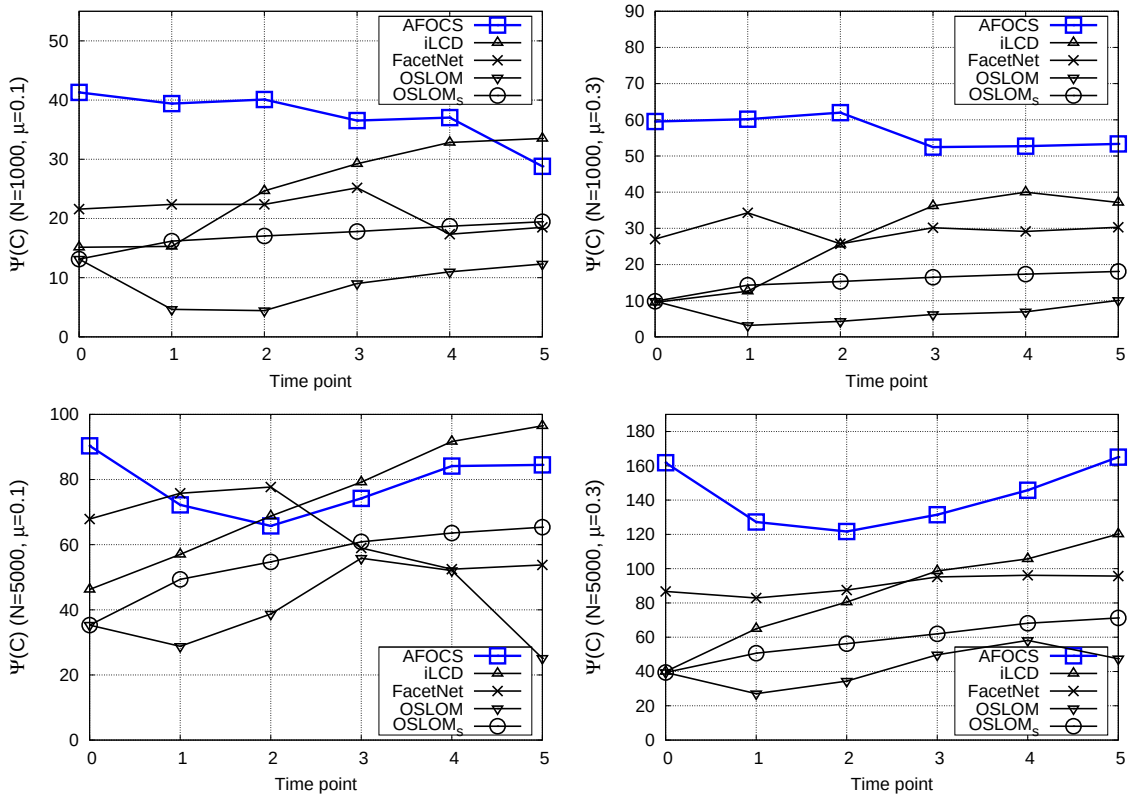
than those of the ground truth but with much higher overlapping rates. The reason is with a larger mixing rate μ, a node will have more edges connecting vertices in other communities, thus increases the chance that AFOCS will merge highly overlapped communities. Hence, AFOCS creates less but with larger size communities. We note that this 'weakness' of AFOCS is controversial as when the mixing rate increases, the ground truth does not necessarily coincide with the structure implied by the network's topology. Extensive experiments show the ability of AFOCS in identifying high quality overlapping communities. In addition, we found AFOCS runs substantially faster than the other competitors: on the Facebook regional network [100] containing 63K nodes, AFOCS is 150x faster than COPRA while CFinder is unable to finish its tasks.

3.4.3 Reference to other dynamic methods

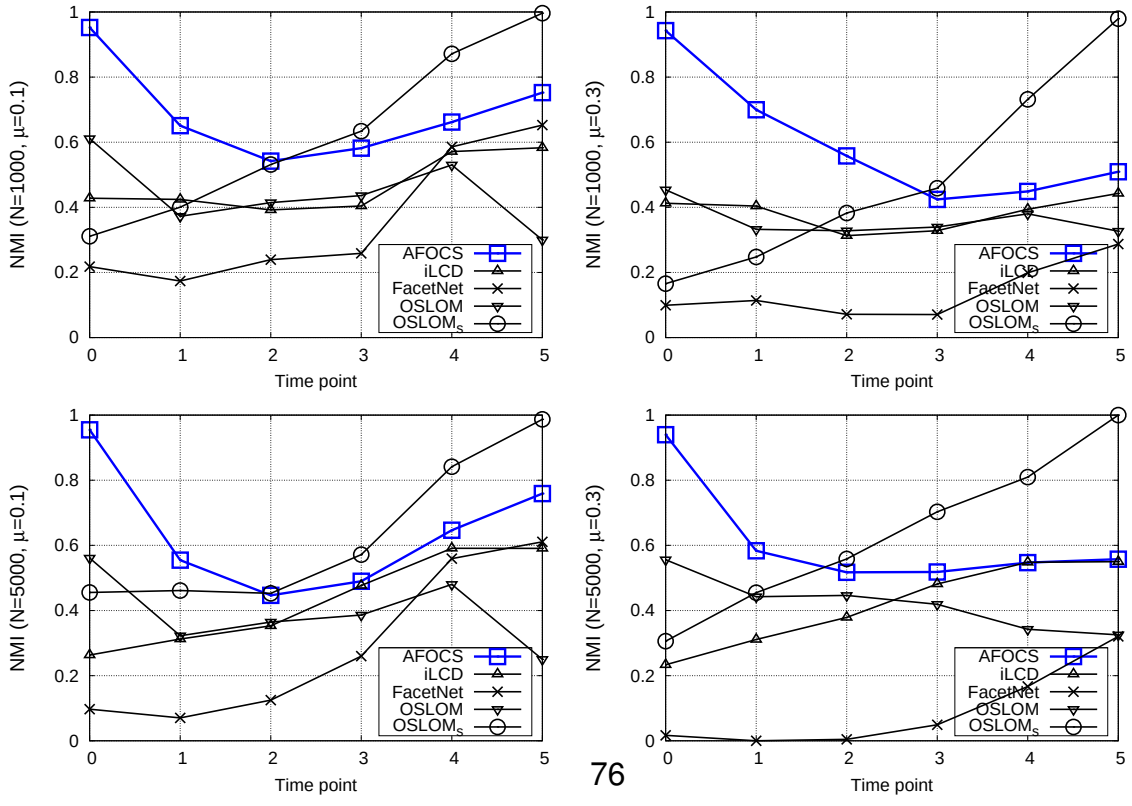
We next observe the performance of AFOCS in reference to two dynamic methods FacetNet, iLCD and OSLOM. Since the ground-truth communities are known on synthesized datasets, fair comparisons among three methods can be obtained via their NMI scores and running times. Of course, the higher its NMI scores with less time consuming, the better the method seems to be.

Each synthesized dynamic network is simulated via 5 snapshots, in which the basic communities are formed by using 50% of the network data with approximately 10% of the network evolution (node/edge additions and removals) added to each growing snapshot at a time. Since FacetNet requires the number of communities a priori, we input this method the actual number as mined form the ground-truth. For iLCD and OSLOM methods, we keep the default setting as provided in their deliverable.

We first evaluate the objective function, i.e., the total internal density obtained by all methods in Figure 3-9A. Although internal density is not necessarily the objective of other methods, this metric can provide us the concept of how strong the community structure detected by each approach is. As revealed Figure 3-9A, AFOCS obtained the highest internal density in all tests and is only lagged behind iLCD approach.



A Objective values



B NMI scores

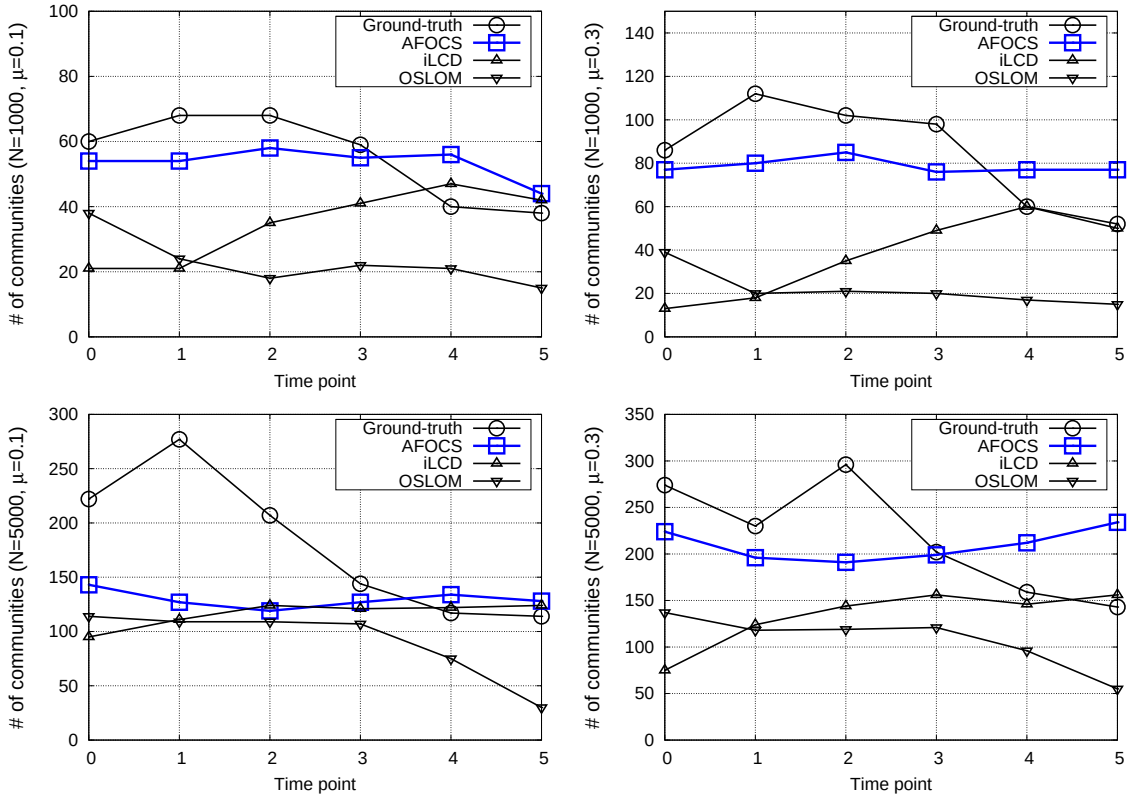


Figure 3-10. The number of communities obtained by AFOCS, iLCD, FacetNet and OSLOM and OSLOM$_s$ methods.

The NMI scores of four methods are presented in Figure 3-9B and 3-10. It reveals from these subfigures that the NMI scores of AFOCS are higher than those of FacetNet, iLCD and OSLOM. In particular, the NMI scores of AFOCS are about just 5-7% lag behind that of OSLOM and iLCD in the first 2 network snapshots, while are much better than the others at the end of the evolution. OSLOM's NMI values are very high at the very beginning, however, they tend to decrease quickly as more connections and nodes are introduced. The NMI scores of iLCD and FacetNet tend to fluctuate and also decrease significantly at the last snapshot. AFOCS, in the other trend, keeps its NMI scores high and wealthy, especially at the end of the network evolution. This implies communities discovered by AFOCS are of higher similarity to the ground-truth than

the other dynamic methods, especially in the long run. The number of communities found by all methods are reported in Figure 3-10. Of course, the closer these detected numbers of communities to the ground-truth, the better the method are believed to be. As revealed in the subfigures of Figure 3-10, these quantities discovered by AFOCS tend to closely approach the actually numbers, even when the mixing rates are high (right figures). The highest similarity between these numbers of communities is possibly the best explanation for the high NMI scores of AFOCS over the other competitors.

We next take a look at the running time of all methods in these synthesized networks. AFOCS requires at most 5 seconds to finish updating each network snapshot whereas FacetNet asks for more than 25 seconds (5x more time consuming) in the networks with just 5000 nodes. iLCD and OSLOM also perform fast in these generated datasets; however, the similarity of the detected communities and the ground-truth is surprisingly poor, as revealed from the results. Therefore, in terms of dynamic approaches, we strongly believe that AFOCS achieves competitive community detection results in a timely manner. These results also provide us the confidence when applying AFOCS to analyze real-world networks.

CHAPTER 4
COMMUNITY STRUCTURE DETECTION USING NONNEGATIVE MATRIX FACTORIZATION

In this chapter, analyze two approaches, namely iSNMF and iANMF, for effectively identifying social network communities using Nonnegative Matrix Factorization (NMF) with I-divergence (Kullback-Leibler divergence) as the cost function. Our approaches work by iteratively factorizing a nonnegative input matrix through derived multiplicative update rules and the Quasi-Newton method. By doing so, we can not only extract meaningful overlapping communities via soft community assignments produced by NMF but also nicely handle both directed and undirected networks with or without weights. We give the complete multiplicative update rules for factorizing $X \approx HH^{\top}$ (iSNMF problem) and $X \approx HSH^{\top}$ (iANMF problem) to effectively identify overlapping communities on social networks. These approaches are topology-independent and their solutions can be easily interpreted. We provide in detail the foundation properties as well as the proofs of correctness and convergence of both iSNMF and iANMF problems. We also propose the Quasi-Newton method to speed up the performance of iSNMF update rule. Furthermore, we validate the performance of our approaches through extensive experiments on not only synthesized datasets but also real-world networks. Empirical results show that iSNMF is among the best efficient detection methods on undirected networks while iANMF outperforms current available methods in directed networks, especially in terms of detection quality.

4.1 Problem Definition and Properties

4.1.1 Motivation for NMF in community detection

Let us first get some insight about how NMF can be helpful in detecting network communities, especially overlapping ones. Consider the toy network $G = (V, E)$ pictured in Figure 4-1. This network contains clear communities C_1 and C_2 having node 4 in common. The adjacency matrix X of this ideal network can be represented as

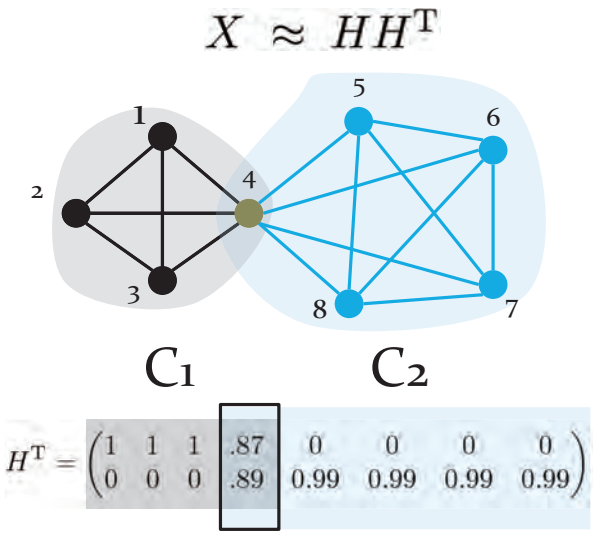


Figure 4-1. An illustrative example motivating NMF in community detection

$X = \begin{pmatrix} S_1 & 0 \\ 0 & S_2 \end{pmatrix}$, where S_1 and S_2 are 4×4 and 5×5 square matrices corresponding to C_1 and C_2, respectively. This adjacent matrix X summarizes all the network information and is the only thing we have. So, how can we derive back the appropriate communities (or the community indicators) only from this matrix? This is where NMF comes into the picture and helps. In particular, the special NMF factorization $X \approx HSH^{\mathrm{T}}$ gives us H and S as the community indicator and the community internal-strength indicator matrices, respectively. In this example, $X \approx HSH^{\mathrm{T}}$ factorization realizes $S = I_2$ and

$$H^{\mathrm{T}} = \begin{pmatrix} 1 & 1 & 1 & .87 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & .89 & 0.99 & 0.99 & 0.99 & 0.99 \end{pmatrix}$$

Matrix H clearly indicates that nodes 1-3 should be in a community and nodes 5-8 should belong to another one. H also advises that node 4 should be an overlapping node due to its significant contribution to both communities. These assignments indeed reflect the true nodes' labels. In addition, matrix S indicates that each detected community attains its perfect internal strength, which intuitively agrees with the original clique structures. This illustrative example, though simple, motivates the application

of the NMF factorization $X \approx HSH^\top$ in community detection. Note that when X is symmetric (i.e., the network is undirected), S is also symmetric and thus, can be further absorbed into H by the assignment $H \leftarrow HS^{1/2}$. Hence, the problem is reduced to $X \approx HH^\top$ only when X is symmetric.

4.1.2 Problem definitions

In order to quantify the goodness of the approximation, we use the I-divergence (Kullback-Leibler (KL) divergence) between two nonnegative matrices A and B suggested by [64] as

$$D(A||B) = \sum_{ij} \left(A_{ij} \log \frac{A_{ij}}{B_{ij}} - A_{ij} + B_{ij} \right)$$

Due to the inequality $x \log x \geq x - 1 \; \forall x > 0$, it is easy to see $D(A||B)$ is lower bounded by zero and vanishes if and only if $A = B$. However, unlike the Euclidean distance, this function is not symmetric in A and B, so we refer to it as the "divergence" from A to B. The smaller the divergence between A and B, the more similar they are. Therefore, our objectives seek for the factorizations $X \approx HH^\top$ and $X \approx HS^\top H$ such that $D(X||HH^\top)$ and $D(X||HSH^\top)$ are minimized. Formally, the problems we are interested in can be stated as follows (here the little "i" comes from the I-divergence)

Problem 1 (iSNMF) Given a nonnegative **symmetric** matrix X, find a matrix $H \geq 0$ that minimizes $D_X(HH^\top) \equiv D(X||HH^\top)$

Problem 2 (iANMF) Given a nonnegative **asymmetric** matrix X, find matrices $H, S \geq 0$ that minimize $D_X(HSH^\top) \equiv D(X||HSH^\top)$

4.1.3 Properties of iSNMF and iANMF factorizations

By Lemma 13, we give important properties of iSNMF and iANMF: the divergences $D_X(HH^\top)$ and $D_X(HSH^\top)$ are convex in S only or H only; however, they are not convex in both variables together. Although the same observations have been proposed for the general NMF problem on both Frobenius and I-divergence cost functions [64], no

claim has been made particularly for the iSNMF and iANMF problems, especially on the I-divergence function.

Lemma 13. *The divergences $D_X(HH^T)$ and $D_X(HSH^T)$ in iSNMF and ANMF are convex in H or S only but not in both S and H together.*

Proof. (Convexity in S) Suppose H is a fixed matrix. For any number $\alpha, \beta \in [0, 1]$ and $\alpha + \beta = 1$, we have

$$D_X(H(\alpha S_1 + \beta S_2)H^T) \leq \alpha D_X(HS_1H^T) + \beta D_X(HS_2H^T),$$

if and only if

$$-\sum_{ij} \log\left(\alpha[HS_1H^T]_{ij} + \beta[HS_2H^T]_{ij}\right) \leq -\alpha \sum_{ij} \log[HS_1H^T]_{ij} - \beta \sum_{ij} \log[HS_2H^T]_{ij}$$

for any matrices $S_1, S_2 \geq 0$. The later inequality holds true due to the convexity of $-\log()$ function and Jensen's inequality. Thus, $D_X(HSH^T)$ is convex in S when H is fixed.

(Convexity in H) Assume S is a fixed matrix. Rewrite

$$D_X(HSH^T) = \sum_{ij} X_{ij}(\log X_{ij} - 1) - \sum_{ij} X_{ij} \log[HSH^T]_{ij} + \sum_{ij} [HSH^T]_{ij}$$

. Since the first term is a constant and $-\log()$ is a convex function, we need to show that the last term is also convex in H. Let $f(H) = \sum_{ij}[HSH^T]_{ij}$. Now,

$$\begin{aligned}\alpha f(H_1) + \beta f(H_2) - f(\alpha H_1 + \beta H_2) &= \alpha\beta \sum_{ij} \left([H_1SH_1^T]_{ij} - [H_2SH_1^T]_{ij} - [H_1SH_2^T]_{ij} + [H_2SH_2^T]_{ij}\right) \\ &= \alpha\beta \sum_{ij} [(H_1 - H_2)S(H_1 - H_2)^T]_{ij} \geq 0\end{aligned}$$

(since $S \geq 0$ and $\sum_{ij}[AA^T]_{ij} \geq 0$ for any matrix A). This implies the convexity in H of $D_X(HSH^T)$.

The convexity of H in iSNMF is derived similarly as above when S is similar to I, the identity matrix. The nonconvexity in both S and H follows from the general NMF case [64]. □

The above properties are nontrivial since they tell us it is unrealistic to solve either iSNMF or iANMF problem for global minima, and consequently give us the hope to use other techniques such as Project Gradient [14], Quasi-Newton [108] or particularly the Alternating Lease Square (ALS) [15] methods to quickly find a local minima. However, by Lemma 14, we show that for iSNMF and iANMF problems, employing the traditional ALS does not provide any speed up since we can neither independently update the columns of S nor H at the same time, thus prevent the employment of this technique to our problems.

Lemma 14. *Employing ALS method does not provide any speed up to either iSNMF or iANMF.*

Proof. Let us first review the ALS method's working mechanism on the general NMF problem $X \approx WH$. Given $X \geq 0$, the ALS method does the following steps [5]

1. Randomly initialize $W^1_{ia} \geq 0, H^1_{bj} \geq 0, \quad \forall i, a, b, j$
2. For $k = 1, 2, \ldots$ alternatively update W^{k+1} and H^{k+1} by $W^{k+1} = \arg\min_{W \geq 0} D_X(WH^k)$, and $H^{k+1} = \arg\min_{H \geq 0} D_X(W^{k+1}H)$;

The main idea of the ALS method is to solve each minimization problem as the collection of several non-negative independent least square problems, due to the uncorrelated relationship between W and H. For instance, one can write $H^{k+1} = \arg\min_{H \geq 0} D(X||W^{k+1}H)$ as H^{k+1}'s jth column $= \min_{\mathbf{h} \geq 0} D(\mathbf{x}||W^{k+1}\mathbf{h})$, where $\mathbf{x}$ is the jth column of X and $\mathbf{h}$ is a column vector of appropriate size. Therefore, each sub-minimization problem requires only the values of a specific column and consequently can be done in a parallel manner. Since H and $H^\top$ are strongly related, it is inappropriate to apply ALS method to iSNMF problem. For $ANMF$ problem, we first note that $[HSH^\top]_{ij} = \sum_{tk} H_{ik}H_{jt}S_{kt}$, which implies an entry in $HSH^\top$ already requires all values of S even when H is fixed. Therefore, should one try to update a single column of S independently as suggested in S-phase of the ALS method, he has to repeatedly

solve for all elements S_{kt}'s, which may incur even more computational requirements. Thus, the conclusion follows. □

4.2 The Update Rule for iSNMF

4.2.1 Multiplicative update rule

We present our solution for iSNMF when the input matrix X is symmetric. Formally, given a nonnegative symmetric matrix X of size $n \times n$ and an integer number $K \ll n$, we need to find a nonnegative matrix H of size $n \times K$ such that $D_X(HH^\top) \equiv D(X||HH^\top)$ is minimized.

We solve this problem using the Karush-Kuhn-Tucker (KKT)[13] conditions. In particular, we introduce the Lagrange multipliers α_{ij} for the constraints $H_{ij} \geq 0$ and consider the objective function $J = D(X||HH^\top) - \sum_{ij} \alpha_{ij} H_{ij}$, or,

$$J = \sum_{ij} \left(X_{ij} \log \frac{X_{ij}}{[HH^\top]_{ij}} - X_{ij} + [HH^\top]_{ij} \right) - \sum_{ij} \alpha_{ij} H_{ij}$$

The KKT conditions require

$$\frac{\partial J}{\partial H_{ab}} = 0 \text{ (or } \frac{\partial D_X(]HH^\top)}{\partial H_{ab}} = \alpha_{ab})$$

as the optimality condition and

$$\alpha_{ab} H_{ab} = 0$$

as a complementary slackness condition for any H_{ab}.

For the ease of computation, we construct the derivative matrix $HH^\top$ with respect to H_{ab} in Figure 4-2. For each position (a, b), this derivative matrix is zero elsewhere except for the a^{th} column and a^{th} row whose elements are from the b^{th} column of H. Using this matrix, we obtain

$$\frac{\partial D_X(HH^\top)}{\partial H_{ab}} = 2 \left(\sum_k H_{kb} - \sum_k H_{kb} \frac{X_{ak}}{[HH^\top]_{ak}} \right). \quad (4\text{–}1)$$

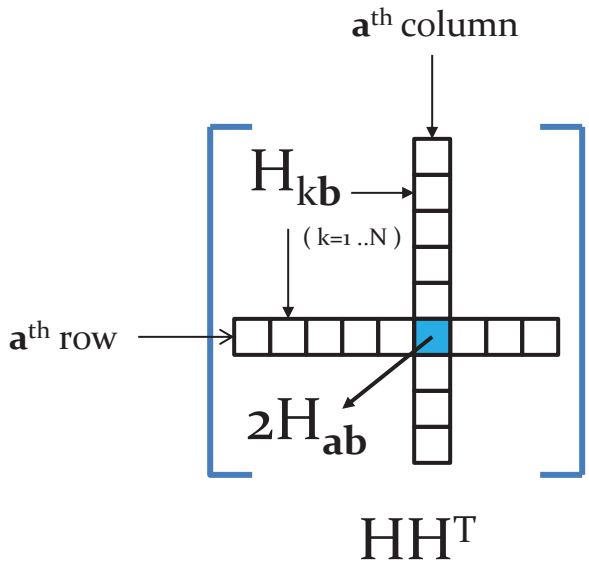


Figure 4-2. The partial derivative matrix of $HH^{\top}$ with respect to H_{ab}.

Hence, the optimality condition implies

$$\alpha_{ab} = 2\Big(\sum_k H_{kb} - \sum_k H_{kb}\frac{X_{ak}}{[HH^{\top}]_{ak}}\Big),$$

and thus, the complementary slackness condition requires

$$2\Big(\sum_k H_{kb} - \sum_k H_{kb}\frac{X_{ak}}{[HH^{\top}]_{ak}}\Big)H_{ab} = 0,$$

which suggests the following update rule

$$H_{ab} \leftarrow H_{ab}\frac{\sum_k H_{kb}X_{ak}/[HH^{\top}]_{ak}}{\sum_t H_{tb}}. \tag{4–2}$$

In terms of projected gradient method, the rule above can be obtained by using the update rule

$$H_{ab} \leftarrow H_{ab} - \nu_{ab}\frac{\partial D_X(HH^{\top})}{\partial H_{ab}},$$

with the magnitude ν_{ab} set to some appropriate small positive number. Here, setting

$$\nu_{ab} = \frac{H_{ab}}{2\sum_t H_{tb}}$$

leads to the same update rule as (4–2).

The iSNMF community detection algorithm is described in Alg. 13. Here, n_0 is the maximum number of iterations, ϵ is the allowed threshold for the quality of iSNMF approximation and α is a given scale to determine community memberships. We assume that K, the number of communities, is predetermined or given as part of the input. Also, the choice of α will be described later.

Algorithm 13 SNMF for community detection

Input: Undirected, unweighted (weighted) adjacent matrix X, K, n_0, ϵ, α;
Output: Community indicator matrix H;

Initialize H to be a random nongnegative matrix;
$iter \leftarrow 0$;
while $(iter \leq n_0)$ and $(D_X(HH^{\mathsf{T}}) > \gamma)$ **do**
    Update $H_{ab} \leftarrow H_{ab} \frac{\sum_k H_{kb} X_{ak} / [HH^{\mathsf{T}}]_{ak}}{\sum_t H_{tb}}$;
    $iter \leftarrow iter + 1$;
end while
% Inferring community labels from H%
$C_b \leftarrow \emptyset \; \forall b = 1 \ldots K$;
$P \leftarrow normalized(H)$;
for $b \leftarrow 1 \ldots p$ **do**
    if $P(a, b) \geq \alpha * max(P(a, :))$ **then**
        $C_b \leftarrow C_b \cup \{a\}$;
    end if
end for

Remark

In contrast to those update rules found in [64], we have shown an important fact: These rules can be derived similarly for this special case. However, our multiplicative update rule (4–2) is not trivial in the sense that we can obtain the convergence proof for our proposed rule whereas one may find it inappropriate to adapt the proof of [64][16] which assumed absolutely no correlation between W and H.

Analysis

We provide the convergence analysis for our proposed update rule (4–2) using an auxiliary function defined as follow:

(Auxiliary function) $G(h, \widetilde{h})$ is the auxiliary function for $F(h)$ if the conditions $G(h, \widetilde{h}) \geq F(h)$ and $G(h, h) = F(h)$ are satisfied

Lemma 15. *[64] If G is an auxiliary function, then F is nonincreasing under the update $h^{t+1} = \arg\min_h G(h, \widetilde{h})$.*

To prove the convergence of the proposed multiplicative update rule, we construct an auxiliary function $G(H, \widetilde{H})$ of $F(H) \equiv D_X(HH^\top)$ as follow

$$G(H, \widetilde{H}) = \sum_{ij} \left(X_{ij} \log X_{ij} - X_{ij} + [HH^\top]_{ij}\right) - \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log H_{ik}H_{jk} - \log \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}\right)$$

Theorem 4.1. *The divergence $D_X(HH^\top)$ is nonincreasing under the update rule (4–2) and is invariant when H is at its stationary point of the divergence.*

Proof. When $\widetilde{H} = H$, it is easy to verify that $G(H, H) = F(H)$, thus we only need to check $G(H, \widetilde{H}) \geq F(H)$. Now, $G(H, \widetilde{H}) \geq F(H)$ iff

$$\begin{aligned}
&- \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log H_{ik}H_{jk} - \log \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}\right) \\
&\geq - \sum_{ij} X_{ij} \log [HH^\top]_{ij} = - \sum_{ij} X_{ij} \log \left(\sum_k H_{ik}H_{jk}\right) \\
\iff &- \sum_{ijk} X_{ij} \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}} \left(\log \frac{H_{ik}H_{jk} \times \sum_t H_{it}\widetilde{H}_{jt}}{H_{ik}\widetilde{H}_{jk}}\right) \\
&\geq - \sum_{ij} X_{ij} \log \left(\sum_k H_{ik}H_{jk}\right).
\end{aligned}$$

To prove the above inequality, we apply Jensen's inequality to the convex function $-\log\left(\sum_k H_{ik}H_{jk}\right)$, yielding

$$-\log \sum_k \alpha_k \frac{H_{ik}H_{jk}}{\alpha_k} \leq - \sum_k \alpha_k \log \frac{H_{ik}H_{jk}}{\alpha_k},$$

where $\alpha_k \equiv \alpha_{ijk} = \frac{H_{ik}\widetilde{H}_{jk}}{\sum_t H_{it}\widetilde{H}_{jt}}$. It is obvious that α_k's are nonnegative and sum up to unity. Thus, we have $G(H, \widetilde{H}) \geq F(H)$. Taking the derivative of $G(H, \widetilde{H})$ with respect to H also gives the same update rule (4–2). □

4.2.2 Quasi-Newton method for iSNMF

One of the problems with the multiplicative update rule is its slow convergence, i.e., it does converge to (possible) stationary point but may be slow, taking more iterations and time, as well as easily getting into local minima trap [5]. One way to speed up the convergence is to adjust the learning rate in a sequential manner, using the second-order estimate of the objective function, e.g. the Quasi-Newton method. In [108], the authors already addressed this method for the general NMF $X \approx WH$ but with the uncorrelated relationship assumption between W and H. Obviously, that assumption does not hold when $X \approx HH^\top$ and hence, it is not trivial to derive proper Quasi-Newton formulation for iSNMF problem. In fact, we show that the second-order, or Hessian, matrix $\mathcal{H}_{D_X}^{(H)}$ of iSNMF is much different from that of the general NMF.

The general Quasi-Newton method, when applied to iSNMF problem, takes the form

$$H \leftarrow \max\left\{H - [\mathcal{H}_{D_X}^{(H)}]^{-1}\frac{\partial D_X}{\partial H}, \epsilon\right\}, \tag{4–3}$$

where D_X is short for $D_X(HH^\top)$, $\frac{\partial D_X}{\partial H}$ is the $n \times K$ first-order matrix of $D_X(HH^\top)$ w.r.t H, $\mathcal{H}_{D_X}^{(H)}$ is the $nK \times nK$ second-order derivative (or Hessian) matrix of D_X w.r.t to H and ϵ is a small nonnegative number to enforce the nonnegativity of H. Thanks to equation (4–1), the first-order derivative matrix $\frac{\partial D_X}{\partial H}$ can be found as

$$\frac{\partial D_X}{\partial H} = 2\left(\mathbf{1} - X./HH^\top\right)H,$$

where $\mathbf{1}$ is a $N \times N$ matrix of all 1's and $./$ is the Hadamard (element-wise) division. For any pair (i, j), the Hessian matrix $\mathcal{H}_{D_X}^{(H)}$ can be found by: $[\mathcal{H}_{D_X}^{(H)}]_{ij} = \frac{\partial D_X}{\partial H_{ij} H_{ab}} =$

$$\begin{cases} 2\left(1 - \frac{X_{ii}([HH^\top]_{ii} - 2H_{ij}^2)}{[HH^\top]_{ii}^2} + \sum_{k\neq i}\frac{H_{ik}^2 X_{ik}}{[HH^\top]_{ik}^2}\right) & a = i, b = j \\ 2\left(\frac{2H_{ib}H_{ij}X_{ii}}{[HH^\top]_{ii}^2} + \sum_{k\neq i}\frac{H_{kb}H_{kj}X_{ik}}{[HH^\top]_{ik}^2}\right) & a = i, b \neq j \\ 2\left(1 - \frac{X_{ak}([HH^\top]_{ai} - H_{ij}H_{aj})}{[HH^\top]_{ai}^2}\right) & a \neq i, b = j \\ 2\left(\frac{H_{ib}H_{aj}X_{ai}}{[HH^\top]_{ai}^2}\right) & a \neq i, b \neq j \end{cases} \tag{4–4}$$

There are two important differences between the Hessian $\mathcal{H}_{NMF}$ for the general case [108] and $\mathcal{H}_{D_X}^{(H)}$. Firstly, $\mathcal{H}_{NMF}$'s elements are zeros everywhere except when $a = i, b = j$ whereas $\mathcal{H}_{D_X}^{(H)}$ obtains values for all combinations of a, b, i and j. Secondly, due to its sparseness, H_{NMF} can be written under matrix block form while $\mathcal{H}_{D_X}^{(H)}$ might not be, particularly when it is a full matrix. Therefore, updating H in iSNMF problem is much more complicated than usual since finding $[\mathcal{H}_{D_X}^{(H)}]^{-1}$ in (4–3) shall require more numerical computations.

The authors in [108] also proposed a numerical technique to overcome the ill-conditioned Hessian matrix and to speed up the computing process, which we find it useful when applied to our problems. Here, we briefly state their technique so that the paper is self-contained (note that (4–5) and (4–6) are not our equations): To reduce the computational cost, the inversion of the Hessian is replaced with the Q-less QR factorization computed by LAPACK. The final form of the Quasi-Newton method is

$$H \leftarrow \max\left\{H - \gamma R_H | W_H, \epsilon\right\} \tag{4–5}$$

$$W_H = Q_H^{\top} \frac{\partial D_X}{\partial H}, \quad Q_H R_H = \mathcal{H}_{D_X}^{(H)} + \lambda I_H \tag{4–6}$$

where I_H is the $nK \times nK$ identical matrix, $\gamma = 10^{-12}$ and $\lambda = 0.9$ are the small fixed regularization and the relax parameters, respectively. The $|$ operator in (4–5) means the Gaussian elimination.

4.3 Update Rules for iANMF

4.3.1 Multiplicative update rules

In this section, we present our solution for the iANMF problem when X is not symmetric. Formally, given a nonegative asymmetric matrix X of size $n \times N$, we find nonnegative matrices H and S of size $n \times K$ and $K \times K$, respectively, such that $D_X(HSH^{\top}) \equiv D(X||HSH^{\top})$ is minimized. We again solve this problem using the KKT conditions by introducing the Lagrange multipliers α_{ij} and β_{ij} for the constraints

$H_{ij} \geq 0$ and $S_{ij} \geq 0$, respectively, and then consider the objective function $J = D(X||HSH^\top) - \sum_{ij} \alpha_{ij} H_{ij} - \sum_{ij} \beta_{ij} S_{ij}$. Equivalently, J can be written as

$$J = \sum_{ij} \left(X_{ij} \log \frac{X_{ij}}{[HSH^\top]_{ij}} - X_{ij} + [HSH^\top]_{ij} \right) - \sum_{ij} \alpha_{ij} H_{ij} - \sum_{ij} \beta_{ij} S_{ij}.$$

The KKT conditions require

$$\frac{\partial J}{\partial H_{ab}} = 0 \text{ and } \frac{\partial J}{\partial S_{ab}} = 0,$$

or equivalently,

$$\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} = \alpha_{ab} \text{ and } \frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \beta_{ab}$$

as the optimality conditions, as well as

$$\alpha_{ab} H_{ab} = 0 \text{ and } \beta_{ab} S_{ab}$$

as a complementary slackness condition for any H_{ab} and S_{ab}. For the ease of computation, we construct the matrix for finding the derivative of an entry $[HSH^\top]_{ij}$ with respect to any H_{ab} in Figure 4-3. Here $r(\mathbf{A}, i)$ and $c(\mathbf{B}, j)$ mean the i^{th} row of A and j^{th} column of B, respectively. Elements outside of the plotted column and row are zeros. The elements of this matrix are zeros elsewhere except for the a^{th} column and a^{th} row. Using this conventional partial derivative matrix, we obtain

$$\frac{\partial \sum_{ij} X_{ij} \log [HSH^\top]_{ij}}{\partial H_{ab}} = \sum_k X_{ka} \frac{[HS]_{kb}}{[HSH^\top]_{ka}} + \sum_k X_{ak} \frac{[SH^\top]_{bk}}{[HSH^\top]_{ak}}$$

$$\text{and} \qquad \frac{\partial \sum_{ij} [HSH^\top]_{ij}}{\partial H_{ab}} = \sum_k ([HS]_{kb} + [SH^\top]_{bk})$$

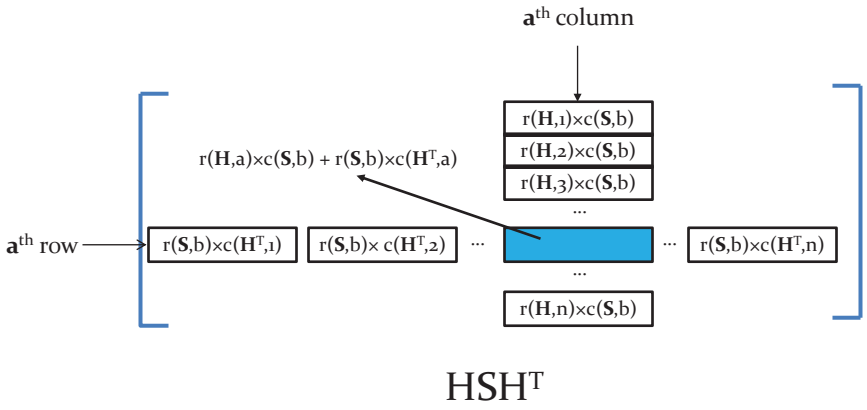


Figure 4-3. The partial derivative matrix of $HSH^\top$ with respect to H_{ab}.

Therefore,

$$\begin{aligned}\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} &= \frac{-\partial \sum X_{ij} \log [HSH^\top]_{ij} + \sum [HSH^\top]_{ij}}{\partial H_{ab}} \\ &= -\sum_k X_{ka} \frac{[HS]_{kb}}{[HSH^\top]_{ka}} - \sum_k X_{ak} \frac{[SH^\top]_{bk}}{[HSH^\top]_{ak}} \\ &\quad + \sum_k ([HS]_{kb} + [SH^\top]_{bk}). \end{aligned} \tag{4–7}$$

The optimality condition $\frac{\partial D_X(HSH^\top)}{\partial H_{ab}} = \alpha_{ab}$ and the complementary slackness condition $\alpha_{ab} H_{ab} = 0$ together give the following update rule for H_{ab}

$$H_{ab} \leftarrow H_{ab} \left(\frac{\sum_k X_{ka}[HS]_{kb}/[HSH^\top]_{ka}}{\sum_t [HS]_{tb} + [SH^\top]_{bt}} + \frac{\sum_k X_{ak}[SH^\top]_{bk}/[HSH^\top]_{ak}}{\sum_t [HS]_{tb} + [SH^\top]_{bt}} \right) \tag{4–8}$$

Alternatively, this update rule can also be achieved by using projected gradient method, in particular by updating

$$H_{ab} \leftarrow H_{ab} - \nu_{ab} \frac{\partial D_X(HSH^\top)}{\partial H_{ab}}$$

with the magnitude

$$\nu_{ab} = \frac{H_{ab}}{\sum_t ([HS]_{tb} + [SH^\top]_{bt})}.$$

Now we give the multiplicative update rule for any S_{ab}. The partial derivative of $D_X(HSH^\top)$ w.r.t S_{ab} is derived as

$$\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \sum_{st} H_{sa}H_{tb} - \sum_{st} X_{st}\frac{H_{sa}H_{tb}}{[HSH^\top]_{st}}$$

The KKT conditions $\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = \beta_{ab}$ and $\beta_{ab}S_{ab} = 0$ together imply the following update rule for S_{ab}

$$S_{ab} \leftarrow S_{ab}\frac{\sum_{st} H_{sa}H_{tb}(X_{st}/[HSH^\top]_{st})}{\sum_{st} H_{sa}H_{tb}} \tag{4–9}$$

Alternatively, this rule can be derived by the projected gradient method

$$S_{ab} \leftarrow S_{ab} - \nu_{ab}\frac{\partial D_X(HSH^\top)}{\partial S_{ab}}$$

with magnitude

$$\gamma_{ab} = \frac{S_{ab}}{\sum_{st} H_{sa}H_{tb}}.$$

The iANMF community detection is presented in Alg. 14. The parameters and their meanings in this case are similar to those described in the $SNMF$ case.

Algorithm 14 iANMF for community detection

Input: Directed, unweighted (weighted) adjacent matrix X, K, n_0, ϵ, α;
Output: Matrices H and S and the inferred community labels;

Initialize H and S to be a random nongnegative matrices;
$iter \leftarrow 0$
while $(iter \leq n_0)$ and $(D_X(HH^\top) > \gamma)$ **do**
    Update H_{ab} based on equation (4–8);
    Update S_{ab} based on equation (4–9);
    $iter \leftarrow iter + 1$;
end while
% Inferring community labels from H%
$C_b \leftarrow \emptyset\ \forall b = 1...K$;
$P \leftarrow normalized(H)$;
for $b \leftarrow 1...K$ **do**
    if $P(a, b) \geq \alpha * max(P(a, :))$ **then**
        $C_b \leftarrow C_b \cup \{a\}$;
    end if
end for

Summary

With the multiplicative update rules (4–8) and (4–9), we give the complete steps for iteratively solving iANMF problem with respect to the I-divergence. These rules are different from what have been discovered in prior studies and, to our knowledge, have not yet been derived in the literature. Thus, they are our contributions in this paper.

Analysis

We first show the following result

Theorem 4.2. *At the stationary point* (H, S) *of* $D_X(HSH^\top)$*, KKT conditions imply that*

$$\sum_{st} X_{st} = \sum_{st} [HSH^\top]_{st}$$

Proof. We show that the condition $S_{ab}\frac{\partial D_X(HSH^\top)}{\partial S_{ab}} = 0$ imply the above equality. In particular, the KKT condition equals to

$$S_{ab}\sum_{st} H_{sa}H_{tb} = S_{ab}\sum_{st} X_{st}\frac{H_{sa}H_{tb}}{[HSH^\top]_{st}}.$$

Summing over all a and b of the LHS gives

$$\sum_{ab} S_{ab}\sum_{st} H_{sa}H_{tb} = \sum_{st}\sum_{ab} H_{sa}S_{ab}H_{tb} = \sum_{st}[HSH^\top]_{st}.$$

Similarly, summing over all a and b of the RHS gives

$$\sum_{ab}\sum_{st} X_{st}\frac{H_{sa}Htb}{[HSH^\top]_{st}} = \sum_{st} X_{st}\frac{[HSH^\top]_{st}}{[HSH^\top]_{st}} = \sum_{st} X_{st}.$$

Therefore, the equality follows. □

We next analyze the convergence analysis of our proposed rules (4–8) and (4–9). By using appropriate auxiliary functions $G(S, \widetilde{S})$ and $G(H, \widetilde{H})$, one can show the following

Theorem 4.3. *The divergence* $D(X||HSH^\top)$ *is nonincreasing under the update rules (4–8) and (4–9) and is invariant if and only if* S *and* H *are at their stationary points in the divergence.*

Proof. The proof of convergence for the two update rules (4–8) and (4–9) is similar to Theorem 4.1. Let us first define two functions

$$G(S, \widetilde{S}) = \sum_{ij} X_{ij}(\log X_{ij} - 1) + \sum_{ij}[HSH^{\top}]_{ij} - \sum_{ij} X_{ij}\beta_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \beta_{ijuv}),$$

$$\text{and } G(H, \widetilde{H}) = \sum_{ij} X_{ij}(\log X_{ij} - 1) + \sum_{ij}[HSH^{\top}]_{ij} - \sum_{ij} X_{ij}\xi_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \xi_{ijuv}).$$

$$\text{where } \beta_{ijuv} = \frac{H_{iv}\widetilde{S}_{vu}H_{ju}}{\sum_{st} H_{it}\widetilde{S}_{ts}H_{js}}, \xi_{ijuv} = \frac{H_{iv}S_{vu}\widetilde{H}_{ju}}{\sum_{st} H_{it}S_{ts}\widetilde{H}_{js}}.$$

It is clear that each β_{ijuv}'s and ξ_{ijuv}'s are nonzero and sum up to unity. We now prove the convergence of rule (4–9) for S when matrix H is fixed. Let $F(S) = D_X(HSH^{\top})$. We show that $G(S, \widetilde{S})$ defined above is an auxiliary function for $F(S)$. When $\widetilde{S} = S$, one can verify that $G(S, \widetilde{S}) = F(S)$, thus we need to check $G(S, \widetilde{S}) \geq F(H)$. This inequality equals to

$$-\sum_{ij} X_{ij} \log [HSH^{\top}]_{ij} \leq -\sum_{ij} X_{ij}\beta_{ijuv}(\log H_{iv}S_{vu}H_{ju} - \log \beta_{ijuv})$$

By the definition of β_{ijuv}, one can rewrite the above inequality as

$$-\log \sum_{ij} \beta_{ijuv} \frac{H_{iv}S_{uv}H_{ju}}{\beta_{ijuv}} \leq -\sum_{ij} \beta_{ijuv} \log \frac{H_{iv}S_{vu}H_{ju}}{\beta_{ijuv}}$$

which generally holds true due to Jensen's inequality and the convexity of $-\log()$ function. Now, taking the derivative of $G(S, \widetilde{S})$ with respect to S gives the update rule (4–9). The proof for H can be obtained in a very similar manner with and thus, is omitted here. □

4.4 Experimental Results

In this section, we first validate our approaches on different synthesized networks with known ground-truths, and then present our findings on real-world traces including the Enron email [98] and Facebook social network [87]. To certify our performance, we

compare the results to two NMF methods proposed in [102] (i.e., wSNMF and wANMF), and the recently suggested Bayesian NMF [90] (i.e., Bayesian method).

Our methods require the number of communities K as an input parameter. We stress that determining this quantity is not the main focus of NMF-based detection methods since almost all of them rely on a predefined K to discover the network communities, as commonly observed in [102][67][70]. Thefore, this quantity K is predetermined using a procedure suggested in [80], which has been shown to well-predict the number of network communities in a timely manner. We also use this value as input for wSNMF and wANMF. For the Bayesian method, we keep the default settings as provided in its deliverable.

4.4.1 Empirical results on synthesized networks

Of course, the best way to evaluate our approaches is to validate them on real-world networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topologies. Although synthesized networks might not reflect all the statistical properties of real ones, they can provide us the known ground-truths via planted communities and the ability to vary other network parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has becomes a usual practice that is widely accepted in the field [58]. Therefore, running iSNMF and iANMF on synthesized networks not only certifies their performance but also provides us the confidence to their behaviors when applied to real-world traces.

Set up: We use the well-known LFR overlapping benchmark [57] to generate 22 weighted directed and undirected testbeds. Generated networks follow the power-law degree distribution and contain embedded overlapping communities of varying sizes that capture the internal characteristics of real-world networks. Parameters are: the number of nodes $N = 1000$, the mixing parameter $\mu = 0.1$ and 0.3 controlling the overall sharpness of the community structure, the weight mixing $\mu_w = 0.1$ and 0.3, the minimum

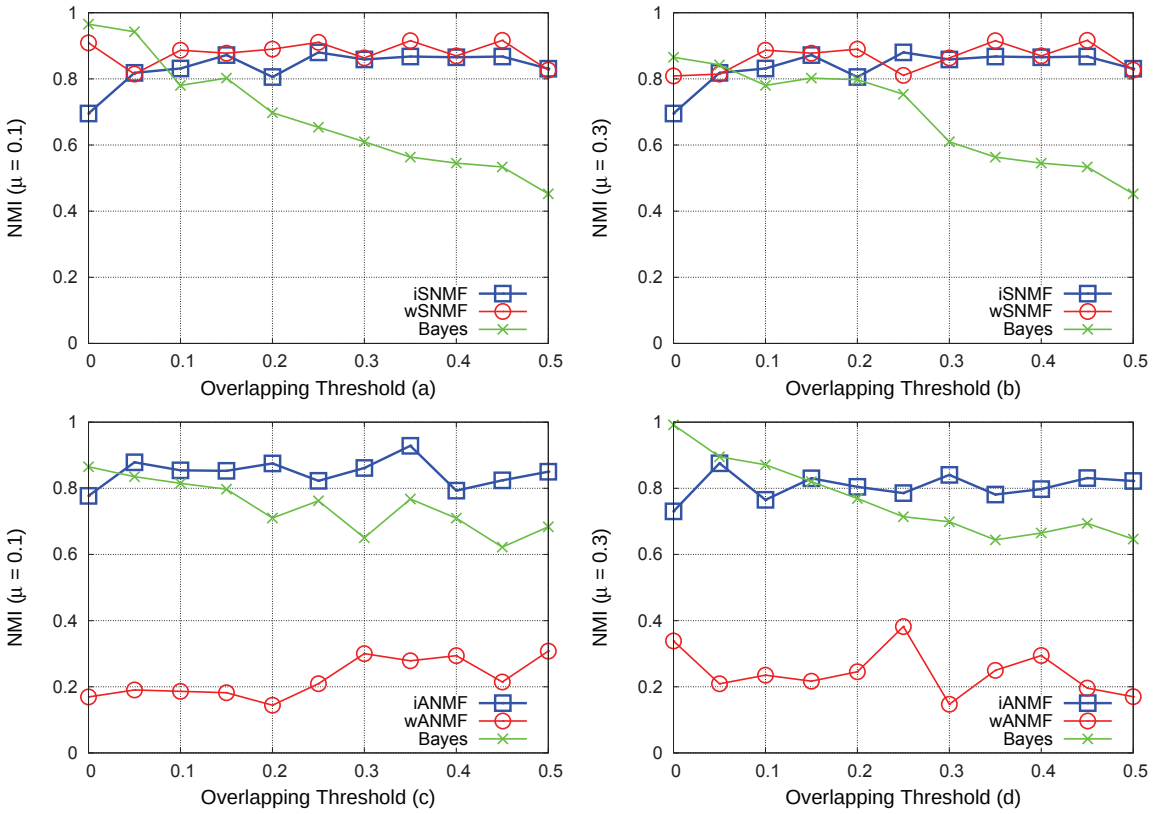


Figure 4-4. Normalized Mutual Information scores on synthesized networks

and maximum community sizes $c_{min} = 10$ and $c_{max} = 50$, the maximum memberships of a node $o_m = 2$, and the overlapping fraction $\gamma \in [0, 0.5]$ measuring the fraction of nodes with memberships in more than communities. We set the number of iterations to 400 in all methods and run 22 tests 100 times for consistency.

Metric: To measure the similarity between detected communities and the embedded ground-truth, we evaluate Generalized Normalized Mutual Information (NMI) [56]. $NMI(U, V)$ is 1 if structures U and V are identical and is 0 if they are totally separated. This is the most important metric for a community detection algorithm because it indicates how good the algorithm is in comparison with the true communities. The higher the NMI value to the ground-truth, the better.

Detection quality: As depicted in Figure 4-4, our approaches iSNMF and iANMF achieve the most stable and competitive (if not to say the best) NMI scores on both

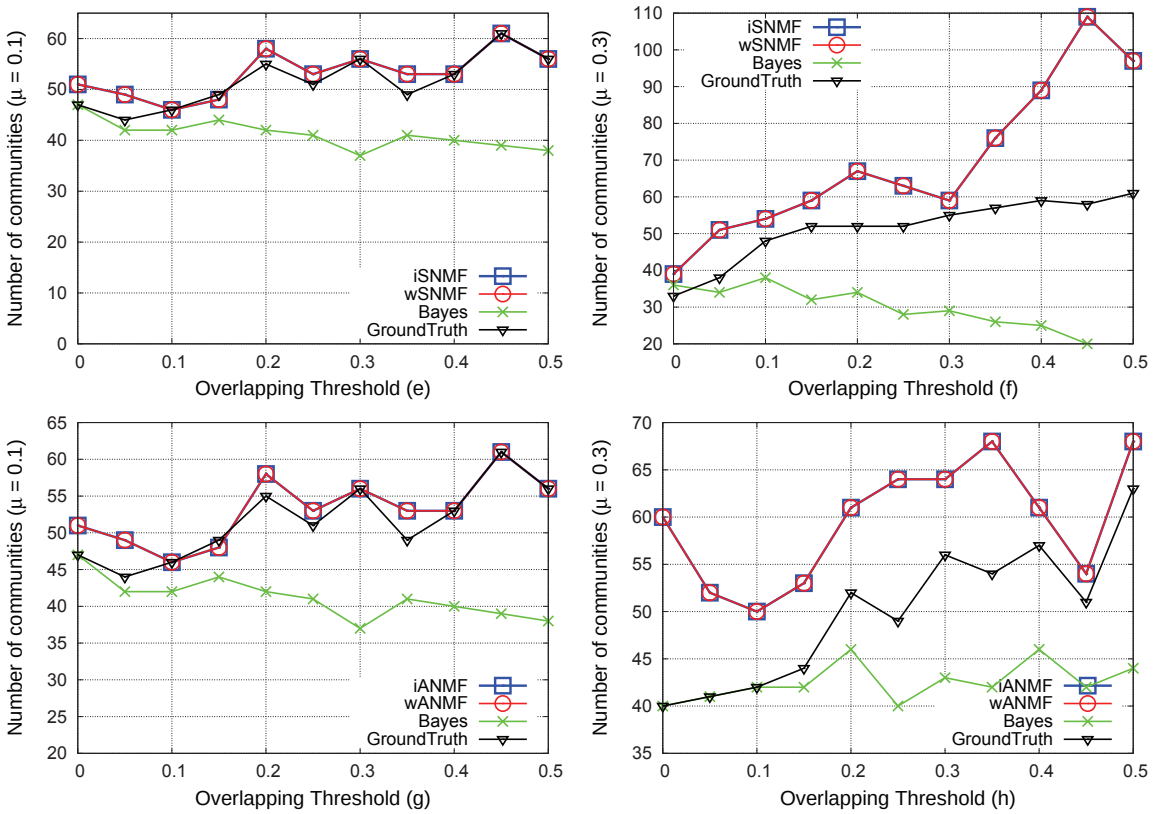


Figure 4-5. Number of communities on synthesized networks

weighted directed and undirected networks. In particular, on undirected networks (top 2 figures), NMI scores produced by iSNMF are highly competitive to those of wSNMF and are up to 84% better than those returned by the Bayesian method. Moreover, its NMI scores still remains high and balance as the mixing overlapping ratio γ increases. This means the communities discovered by iSNMF are consistently of high similarity to the ground-truth even when more and more network communities are overlapped with each other. wSNMF also displays these properties on undirected networks; however, its performance degrades significantly on directed weighted networks, as we will discuss shortly. The Bayesian method, on the other hand, produces very low NMI values that tend to decrease quickly as γ increases. This implies communities detected by this method are not ideally coincident with the embedded ones, especially when they highly overlap with each other.

There is a close relationship between the number of communities and the identification capacity that we observed in the case of undirected networks in Figure 4-5. As revealed in its top figures, the input numbers of communities for iSNMF and iSNMF are almost identical to the ground-truth when $\mu = 0.1$ and slightly deviate from them when $\mu = 0.3$, while those of the Bayesian method are far away from the baseline. This close relationship, as a result, helps iSNMF and wSNMF to determine a proper number of basic features and consequently, indicate more appropriate community labels. However, this observation does not appear to hold for wANMF on directed networks since it performs poorly whereas our approach iANMF still performs excellently on this type of networks (Figure 4-4, bottom figures). The big gap between the Bayesian method and the ground-truth implies its built-in estimate of the number of communities could potentially mislead the factorization, thus results in its low NMI scores.

The superiority of our iANMF approach becomes more visible on directed weighted networks (Figure 4-4, bottom figures). In these figures, iANMF returns the best stable NMI values and they remain wealthy even when γ evolves, i.e., when strongly overlapped communities appear. In particular, the NMI scores returned by iANMF are more than twice those of wANMF and are up to 10% those of Bayesian method. The performance of wANMF, surprisingly, reduces to no more than half of its prior achievement even when fed with the relatively close number of true communities (bottom figures of Figure 4-5). This in turn indicates the communities discovered by wANMF are heavily deviated from and are of very low similarity to the ground-truth. Bayesian method's performance is somehow the same on these directed networks with average NMI scores tend to quickly decrease in the long run. This comparison among three NMF factorizations reveals that iSNMF and iANMF are the best ideal methods for effectively recovering the overlapped network community structures, especially on weighted and directed networks.

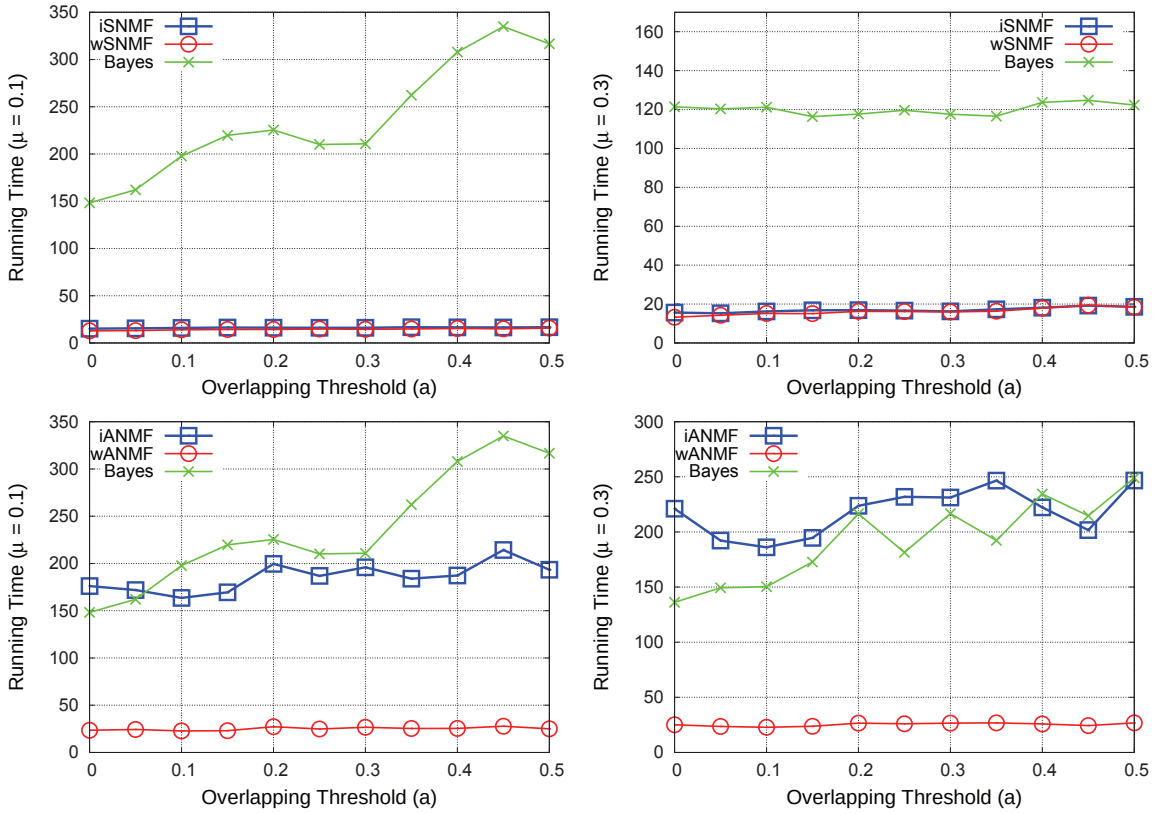


Figure 4-6. Running Time on synthesized networks

We next compare the running time of three methods. As reported in Figure 4-6, the running times of iSNMF and wSNMF on undirected networks are fairly similar to each other (at most 2s difference) and are much less than the huge time requirement of the Bayesian method. In average, the Bayesian method requires almost 200s in order to finish the test whereas iSNMF and wSNMF only ask for roughly 16s and 14s, respectively. On directed networks, iANMF requires nearly the same amount of time of the Bayesian method and much more time than wANMF. Note that this time consumption of iANMF is quite understandable because each update for S_{ab} in equation (4–9) based on the I-divergence already took $O(n^2)$ time. However, the superiority of its produced NMI scores to other competitors makes iANMF a promising approach, especially suited for those who strive to discover excellent network community structures.

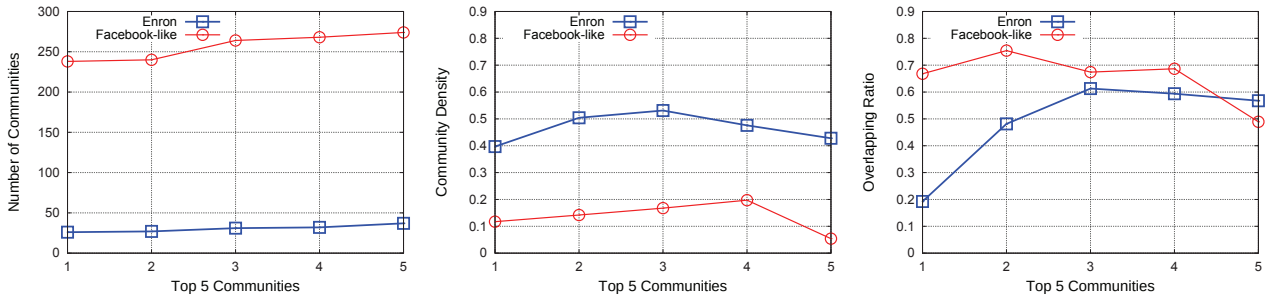


Figure 4-7. The number of communities, Internal density and Overlapping ratio of Enron email and Facebook-like datasets

In summary, comparisons among three algorithms on generated networks show that (1) iSNMF is among the best NMF methods for efficiently identify high quality overlapping communities in weighted and unweighted undirected networks (2) iANMF is the best among three methods for analyzing weighted and directed networks containing highly overlapped communities, despite it long running time. More importantly, the performance of both approaches remains healthily stable even when more and more overlapping communities are introduced. These results provide us the strong confidence when applying iSNMF and iANMF to analyze the real-world traces.

4.4.2 Results on real networks

We next utilize iANMF and iSNMF to analyze the real network datasets and present our findings on their overlapping structures. In particular, we choose the Enron email dataset and the Facebook-like social network [87]. The Enron email network contains email messages data from about 150 users, mostly senior management of Enron Inc., from Jan 1999 to July 2002 [98]. Each email address is represented by an unique identification number in the dataset and each link corresponds to a message sent between the sender and the receiver. The Facebook-like social network is collected from students of University of California, Irvine. The dataset contains 20296 messages sent and received among 1899 users. The number of communities inputed for Enron email and Facebook-like datasets are set to 8 and 18, respectively.

We are interested in understanding their overlapping structures and what the overlapping nodes really mean to them, particularly in the top 5 biggest communities. As revealed in Figure 4-7, the numbers of members in top 5 communities of Facebook-like network are, not surprisingly, much bigger are those of the Enron email network. However, the internal density, i.e., the inner structures of those top 5 communities in Enron emails are much stronger than those of Facebook networks. Indeed, the density values of Enron email communities are more than twice of Facebook networks. This can be explained as email communication in a work place among managers occurs much more frequently than messages on a social environment like the Facebook network.

We next investigate on the overlapping substructures of these real networks, i.e., we want to know how much they are overlapped and what the overlapped nodes mean to the communities. As described in Figure 4-7, all 5 top communities of Facebook network are highly overlapped whereas just 3 top communities of Enron email network appear to have this properties. Moreover, overlapped nodes on Facebook network tend to be active users who eagerly participate in multiple communities at the same time, i.e., they send messages to multiple friends in different groups. Overlapped nodes on Enron email network, though fewer, suggest that they potentially play vital roles in the company since most of them communicate frequently many other members in all of the communities.

CHAPTER 5
SOCIAL-AWARE ROUTING STRATEGIES IN MOBILE AD-HOC NETWORKS

In this chapter, we demonstrate the applicability of our proposed detection algorithms QCA and AFOCS as the community identification cores in forwarding and routing strategies in mobile dynamic networks. In the following paragraphs, we first present the application of QCA and then describe how AFOCS can help to improve the performance of this practical applications.

5.1 A Message Forwarding and Routing Strategy Employing QCA

In a broad view, a MANET is a dynamic wireless network with or without the underlying infrastructure, in which each node can move freely in any direction and organize itself in an arbitrary manner. Due to nodes mobility and unstable links nature of a MANET, designing an efficient routing scheme has become one of the most important and challenging problems on MANETs. Recent researches have shown that MANETs exhibit the properties of social networks [46][19][10] and social-aware algorithms for network routing are of great potential. This is due to the fact that people have a natural tendency to form groups or communities in communication networks, where individuals inside each community communicate with each other more frequent than with people outside. This social property is nicely reflected to the underlying MANETs by the existence of groups of nodes where each group is densely connected inside than outside. This resembles the idea of community structure in Mobile Ad hoc Networks.

Multiple routing strategies [19]-[45] based on the discovery of network community structures have provided significant enhancement over traditional methods. However, the community detection methods utilized in those strategies are not applicable for dynamic MANETs since they have to recompute the network structure whenever changes to the network topology are introduced, which results in significant computational costs and processing time. Therefore, employing an adaptive community structure

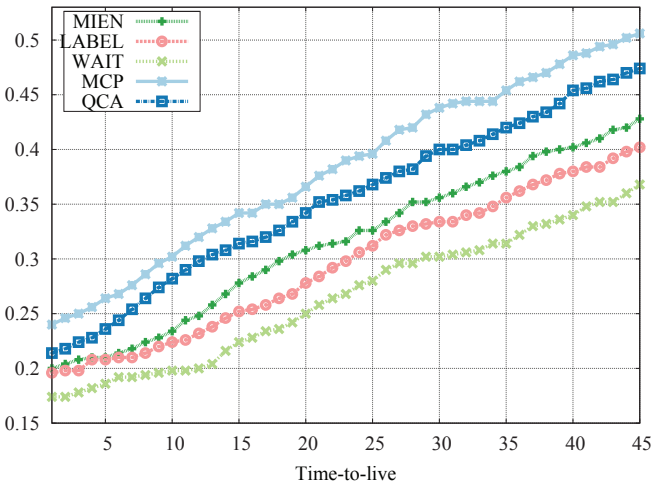
detection algorithm as a core will provide a speedup as well as robust to routing strategies in MANETs.

We evaluate five routing strategies (1) WAIT: the source node waits until it meets the destination node (2) MCP: A node keeps forwarding the messages until they reach the maximum number of hops (3) LABEL: A node forwards or sends the messages to all members in the destination community [46] (4) QCA: A Label version utilizing QCA as the dynamic community detection method and lastly, (5) MIEN: A social-aware routing strategy on MANETs [26].

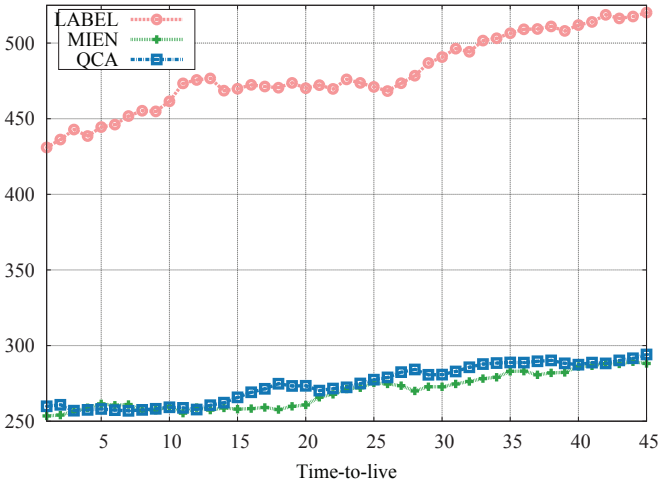
Even though WAIT and MCP algorithms are very simple and straightforward to understand, they provide us helpful information about the lower and upper bounds on the message delivery ratio, time redundancy as well as message redundancy. The LABEL forwarding strategy works as follow: it first finds the community structure of the underlying MANET, assigns each community with the same label and then exclusively forwards messages to destinations, or to next-hop nodes having the same labels as the destinations. MIEN forwarding method utilizes MIEN algorithm as a subroutine. QCA routing strategy, instead of using a static community detection method, employs QCA algorithm for adaptively updating the network community structure and then uses the newly updated structure to inform the routing strategy for forwarding messages.

5.1.1 Setup

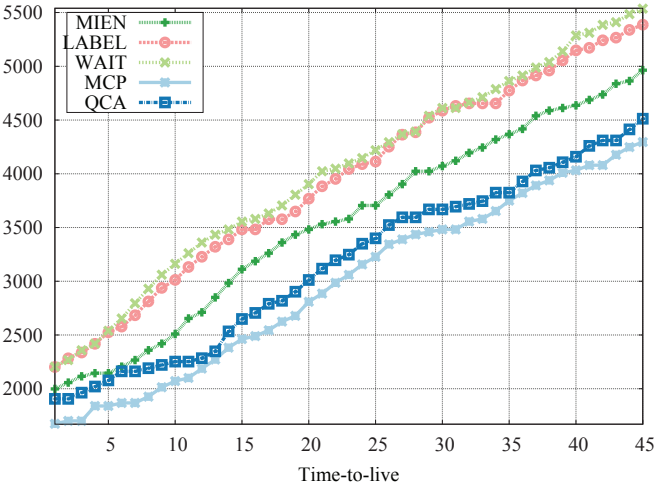
We choose Reality Mining data set [29] provided by the MIT Media Lab to test our proposed algorithm. The Reality Mining data set contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, the data set includes call logs, Bluetooth devices in proximity, cell tower IDs, application usage, and phone status (such as charging and idle) of the participated students of over 350,000 hours ($\tilde{4}$0 years). In this paper, we take into account the Bluetooth information to form the underlying MANET and evaluate the performance of the above five routing strategies.



A Delivery Ratio



B Average Duplicate Message



C Average Delivery Time

Figure 5-1. Experimental results on the Reality Mining data set

5.1.2 Results

For each routing method, we evaluate the followings (1) Delivery ratio: The portion of successfully delivered over the total number of messages (2) Average delivery time: Average time for a message to be delivered. (3) Average number of duplicated messages for each sent message. In particular, a total of 1000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold time-to-live. The experimental results are shown in Figure $5-1A$, $5-1B$ and $5-1C$.

Figure $5-1A$ describes the delivery ratio as a function of time-to-live. As revealed by this figure, QCA achieves much better delivery ratio than MIEN as well as LABEL and far better than WAIT. This means that QCA routing strategy successfully delivers many more messages from the source nodes to the destinations than the others. Moreover, as time-to-live increases, the delivery ratio of QCA tends to approximate the ratio of MCP, the strategy with highest delivery ratio.

Comparison on delivery time shows that QCA requires less time and gets messages delivered successfully faster than LABEL, as depicted in Figure $5-1C$.
It even requires less delivery time in comparison with the social-aware method MIEN. This can be explained as the static community structures in LABEL can possibly get message forwarded to a wrong community when the destinations eventually change their communities during the experiment. Both QCA and MIEN, on the other hand, captures and updates the community structures on-the-fly as changes occur, thus achieves better results.

The numbers of duplicate messages presented in Figure $5-1B$ indicate that both QCA and MIEN achieves the best results. The numbers of duplicated messages of MCP method are substantially higher than those of the others and are not plotted. In fact, the results of QCA and MIEN are relatively close and tend to approximate each other as time-to-live increases.

In conclusion, QCA is the best social-aware routing algorithm among five routing strategies since its delivery ratio, delivery time, and redundancy outperform those of the other methods and are only below MCP while the number of duplicate messages is much lower. QCA also shows a significant improvement over the naive LABEL method which uses a static community detection method and thus, confirms the applicability of our adaptive algorithm to routing strategies in MANETs.

5.2 A Message Forwarding and Routing Strategy Employing AFOCS

We present a practical application where the detection of overlapping network communities plays a vital role in forwarding strategies in communication networks. With the helpful knowledge of the network community structure discovered by AFOCS, we propose a new community-based forwarding algorithm that significantly reduces the number of duplicate messages while maintaining competitive delivery times and ratios, which are essential factors of a forwarding strategy.

5.2.1 Message forwarding strategy

Let us first discuss how our new forwarding algorithm works in practice and then how AFOCS helps it to overcome the above limitations. We use AFOCS to detect overlapping communities and keep it up-to-date as the network changes. Each node in a community is assigned the same label and each overlapped node u has a set of corresponding labels $Com(u)$. During the network operation, if a devices u carrying the message meets another device v who indeed shares more common community labels with the destination than u, i.e., $|Com(v) \cap Com(dest)| > |Com(u) \cap Com(dest)|$, then u will forward the message to v. The same actions then apply to v as well as to devices that v meets.

The intuition behinds this strategy is that if v shares more communities with the destination nodes, it is likely that v will have more chances to deliver the message to the destination. By doing in this way, we not only have higher chances to correctly forward the messages but also generate much less duplicate messages. Due to its

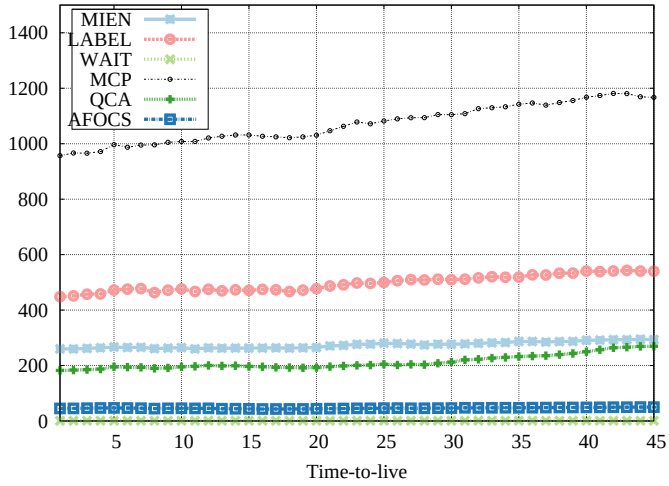
adaptive nature and the ability of identifying overlapping communities, AFOCS helps our algorithm to overcome the above shortcomings naturally. This explains why our forwarding algorithm can significantly reduce the number of duplicate messages while maintaining very competitive delivery times and ratios.

5.2.2 Setup

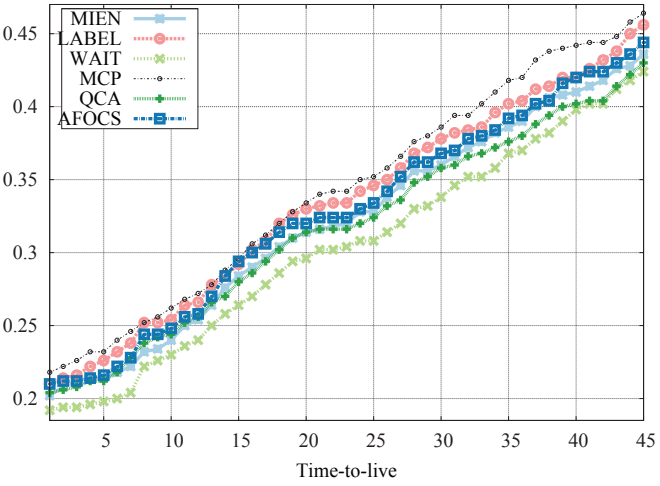
We compare six forwarding strategies (1) MIEN: A recently proposed social-aware routing strategy on MANETs [26] (2) LABEL: A node will forward the messages to another node if it is in the same community as the destination [46] (3) WAIT: The source node waits and keeps forwarding the message until it meets the destination (4) MCP: A node keeps forwarding the messages until they reach the maximum number of hops (5) QCA: A LABEL version utilizing QCA [81] as the adaptive disjoint community detection method and lastly (6) AFOCS: Our newly proposed forwarding algorithm equipped with AFOCS as an community detection and update core.

Results of WAIT and MCP algorithms provide us the lower and upper bounds of important factors: message delivery ratio, time redundancy and message redundancy. Our experiments are performed on the Reality Mining dataset provided by the MIT Media Lab [29]. This dataset contains communication, proximity, location, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. In particular, we take into account the Bluetooth information to construct the underlying communication network and evaluate the performance of the above six routing strategies.

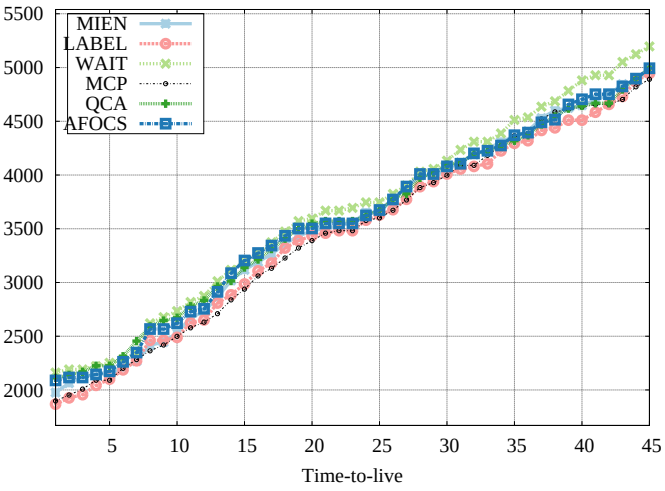
In each experiment, 500 message sending requests are randomly generated and distributed in different time points. To control the forwarding process, we use hop-limit, time-to-live, and max-copies parameters. A message cannot be forwarded more than hop-limit hops in the network or exist in the process longer than time-to-live, otherwise it will be automatically discarded. Moreover, the maximum number of same messages



A Average Duplicate Message



B Delivery Ratio



C Average Delivery Time

Figure 5-2. Experimental results on the Reality Mining data set

a device can forward to the others is restricted by max-copies. Experiments results are repeated and results are averaged for consistency.

5.2.3 Results

Our results are presented in Figures 5-2A, 5-2B, 5-2C. The first observation reveals that our proposed forwarding algorithm achieves the lowest number of duplicate messages as depicted in Figure 5-2A, and even far better than the second best method QCA. On average, only 46.5 duplicate messages are generated by AFOCS during evaluation process in contrast with 212.2 of QCA, 274.2 of MIEN, 496.4 of LABEL and the huge 1071.0 overhead messages of MCP. Thus, on the number of duplicate messages, AFOCS strikingly achieves improvement factors of 4.5x, 5x, 11x and 23x over these mentioned strategies, respectively. These extremely low overhead strongly imply the efficiency of AFOCS in communication networks.

Figures 5-2B and 5-2C present our results on the other two important factors, the message delivery ratios and delivery times. These figures supportively indicate that AFOCS achieves competitive results on both of these vital factors. In general, AFOCS is the second best strategy with almost no noticeable different between itself and the leader method LABEL. On average, AFOCS gets 33% of the total messages delivered in 3569.2s and only a little bit lags over MCP (34% in 3465.3s) and LABEL (slightly over 33% in 3462.7s), and is far better than MIEN (32% in 3537.6s) and QCA (32% in 3572.2s). This can be explained by the advantages of knowing the overlapping community structure: the disjoint network communities in QCA and MIEN can possibly have messages forwarded to the wrong communities when the destination changes its membership. With the ability of quickly updating the network structure, AFOCS can efficiently cope with this change and thus, can still provide the most updated forwarding information.

In summary, AFOCS helps our forwarding strategy to reduce up to 11x the number of duplicate messages while keeping good average delivery ratio and time. These

experimental results are highly competitive and supportively confirm the effectiveness of AFOCS and our new routing algorithm on communication networks.

CHAPTER 6
SOLUTIONS FOR WORM CONTAINMENT IN ONLINE SOCIAL NETWORKS

In this section, we present another practical application of our proposed algorithms in worm containment problem in OSNs. We first suggest a solution based on QCA, and then describe how AFOCS can help to improve the performance of this solution for this practical problem in complex networks. Since their introduction, popular social network sites such as Facebook, Twitter, Bebo, and MySpace have attracted millions of users worldwide, many of whom have integrated those sites into their everyday lives. On the bright side, OSNs are ideal places for people to keep in touch with friends and colleagues, to share their common interests, or just simply to socialize online. However, on the other side, social networks are also fertile grounds for the rapid propagation of malicious softwares (such as viruses or worms) and false information.

Facebook, one of the most famous social sites, experienced a wide propagation of a trojan worm named “Koobface” in late 2008. Koobface made its way not only through Facebook but also Bebo, MySpace and Friendster social networks [31][53]. Once a user’s machine is infected, this worm scans through the current user’s profile and sends out fake messages or wall posts to everyone in the user’s friend list with titles or comments to appeal to people’s curiosity. If one of the user’s friends, attracted by the comments without a shadow of doubt, clicks on the link and installs the fake “flash player”, his computer will be infected and Koobface’s life will then cycle on this newly infected machine.

Worm containment problem becomes more and more pressing in OSNs as this kind of networks evolves and changes rapidly over time. The dynamics of social networks thus gives worms more chances to spread out faster and wider as they can flexibly switch between existing and new users in order to propagate. Therefore, containing worm propagation on social networks is extremely challenging in the sense that a good solution at the previous time step might not be sufficient or effective at the next time

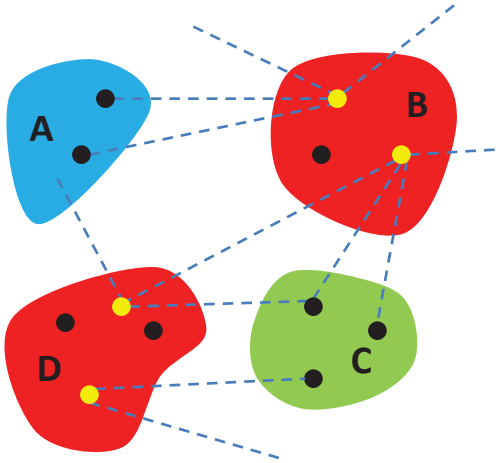


Figure 6-1. A general worm containment strategy.

step. Although one can recompute a new solution at each time the network changes, doing so would result in heavy computational costs and be time consuming as well as allowing worms spreading out wider during the recomputing process. A better solution should quickly and adaptively update the current containing strategy based on changes in network topology, and thus can avoid the hassle of recomputation.

There are many proposed methods for worm containment on computer networks by either using a multi-resolution approach [97], or using a simplification of the Threshold Random Walk scan detector [106], or using fast and efficient worm signature generation [51]. There are also several methods proposed for cellular and mobile networks [104][7]. However, these approaches fail to take into account the community structure as well as the dynamics of social networks, and thus might not be appropriate for our problem. A recent work [110] proposed a social-based patching scheme for worm containment on cellular networks. However, this method encounters the following limitations on a real social network (1) its clustered partitions do not necessarily reflect the natural network communities, (2) it requires the number of clusters k (which is generally unknown for social networks) must be specified beforehand, and (3) it exposes weaknesses when dealing with the network's dynamics.

6.1 An Application of QCA in Containing Worms in OSNs

6.1.1 Setup

To overcome these limitations, our approach first utilizes QCA to identify the network community structure, and adaptively keeps this structure updated as the network evolves. Once network communities are detected, our patch distribution procedure will select the most influential users from different communities in order to send patches. These users, as soon as they receive patches, will apply them to first disinfect the worm and then redistribute them to all friends in their communities. These actions will contain worm propagation to only some communities and prevent it from spreading out to a larger population. To this end, a quick and precise community detection method will definitely help the network administrator to select a more sufficient set of critical users to send patches, thus lowering down the number of sent patches as well as overhead information over the social network.

Algorithm 15 Patch Distribution

Input: $G = (V, E)$ and its community structure $\mathcal{C} = \{C_1, C_2, .., C_p\}$
Output: The set of influential users $\mathcal{P}$.

1: $\mathcal{P} = \emptyset$;
2: **for** $C_i \in \mathcal{C}$ **do**
3: **while** $\exists u$ unvisited in C_i satisfying $\max_{u \in C_i}\{e_{out}^{C_i}(u)\} > 0$ **do**
4: Let $v \leftarrow \arg\max_{u \in C_i}\{e_{out}^{C_i}(u)\}$;
5: $\mathcal{P} = \mathcal{P} \cup v$;
6: Mark v as visited in C_i;
7: **end while**
8: **end for**
9: Send patches to users in $\mathcal{P}$;

We next describe our patch distribution. This procedure takes into account the identified network communities and selects a set of influential users from each community in order to distribute patches. Influential users of a community are ones having the most relationships or connections to other communities. In an adversary point of view, these influential users are potentially vulnerable since they not only interact actively within their communities but also with people outside, and thus, they

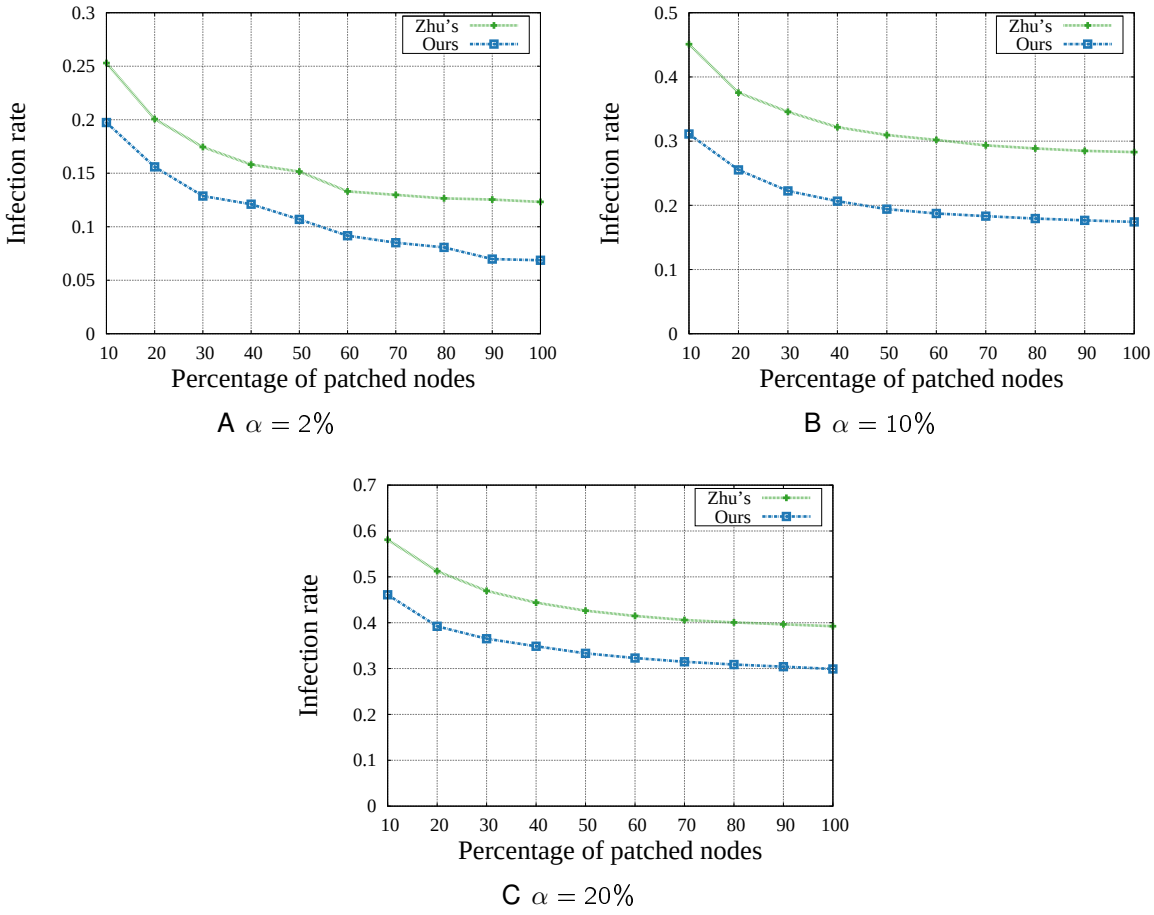


Figure 6-2. Infection rates on static network with $k = 150$ clusters

can easily fool (or be fooled by) people both inside and outside of their communities. On the other point of view, these users are also the best candidates for the network defender to distribute patches since they can easily announce and forward patches to other members and non-members.

In Alg. 15, we present a quick algorithm for selecting the set of most influential users in each community. This algorithm starts by picking the user whose number of social connections to outside communities is the highest, and temporarily disregards this user from the considering community. This process repeats until no connections crossing among communities exists. This set of influential users is the candidate for the network defender for distributing patches.

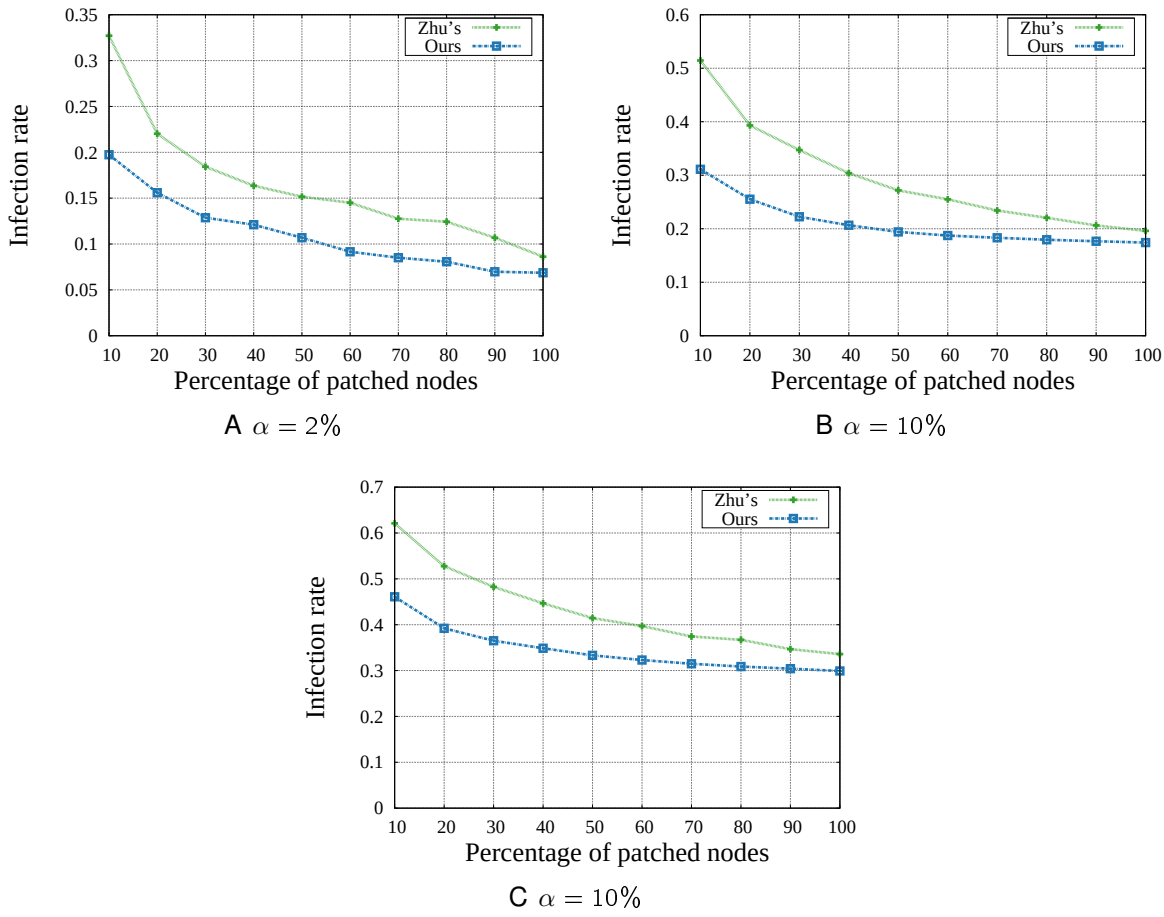


Figure 6-3. Infection rates on dynamic network with $k = 200$ clusters

6.1.2 Results

We present the results of our QCA method on the Facebook network dataset [100] and compare the results with the social based method (Zhu's method [110]) via a weighted version of our algorithms.

The worm propagation model in our experiments mimics the behavior of the famous "Koobface" worm. The probabilities of activating the worm is proportional to communication frequency between the victim and his friends. The time taken for worms to spread out from one user to another is inversely proportional to the communication frequency between this user and his particular friend. Finally, when a worm has successfully infected a user's computer, it will start propagating as soon as

this computer connects to a specific social network (Facebook in this case). When the fraction of infected users reaches a threshold α, the detection system raises an alarm and patches will automatically be sent to most influential users selected by Alg. 15. Once a user receives the patch, he will first apply it to disinfect the worm and then will have an option to forward it to all friends in his community. Each experiment is seeded with 0.02% of users to be initially infected by worms.

We compare infection rates of the social-based method of Zhu's and ours. The infection rate is computed as the fraction of the remaining infected users over all infected ones. The number of clusters k in Zhu's method is set to be 150 in static and 200 in dynamic networks, and for each value of k, the alarming threshold α is set to be 2%, 10%, and 20%, respectively. Each experiment is repeated 1000 times for consistency.

Figure 6-2, 6-3 show the results of our experiments for three different values of k and α. We first observe that the longer we wait (the higher the alarm threshold is), the higher number of users we need to send patches to in order to achieve the desired infection rate. For example, with $k = 150$ clusters and an expected infection rate of 0.3, we need to send patches to less than 10% number of users when $\alpha = 2\%$, to more than 15% number of users when $\alpha = 10\%$ and to nearly 90% of total influential users when $\alpha = 20\%$.

A second observation reveals that our approach achieves better infection rates than the social-based method of Zhu's in a static version of the social network as depicted in Figure 6-2. In particular, the infection rates obtained in our method are from 5% to 10% better than those of Zhu's. When the network evolves as new users join in and new social relationships are introduced, we resize the number of cluster k and recompute the infection rates of the social based method with the number of cluster $k = 200$, and the alarm threshold $\alpha = 2\%$ and 10% respectively. As depicted in Figures 6-3, our method, with the power of quickly and adaptively updating the network community structure, achieves better infection rates than Zhu's method while the computational costs and

running time is significantly reduced. As discussed, detecting and updating the network community is the crucial part of a social based patching scheme: a good and up-to-date network community structure will provide the network defender a tighter set of vulnerable users, and thus, will help to achieve lower infection rates. Our adaptive algorithm, instead of recomputing the network structure every time changes are introduced, quickly and adaptively updates the network communities on-the-fly. Thanks to this frequently updated community structure, our patch distribution procedure is able to select a better set of influential users, and thus, helps in reducing the number of infected users.

We further look more into the behavior of Zhu's method when the number of clusters k varies. We compute and compare the infection rates on Facebook dataset for various k ranging from 1K to 2.5K with our approach. We first hope that the more predefined clusters, the better infection rates clustered partitioning method will achieve. However, the experimental results reveal the opposite. In particular, with a fixed alarming threshold $\alpha = 10\%$ and 60% patched nodes, the infection rates achived by Zhu's method do not decrease but ranging near 28% while ours are far better (20%) with much less computational time.

Finally, a comparison on running time on the two approaches shows that time taken for Zhu's method is much more than our community updating procedure, and hence, may prevent this method to complete in a timely manner. In particular, our approach takes only 3 seconds for obtaining the basic community structure and at most 30 seconds to complete all the tasks whereas [110] requires more than 5 minutes to divide the communication network into modules and selecting the vertex separators. In that delay, worm propagation may spread out to a larger population, and thus, the solution may not be effective. These experimental results confirm the robustness and efficiency of our approach on social networks.

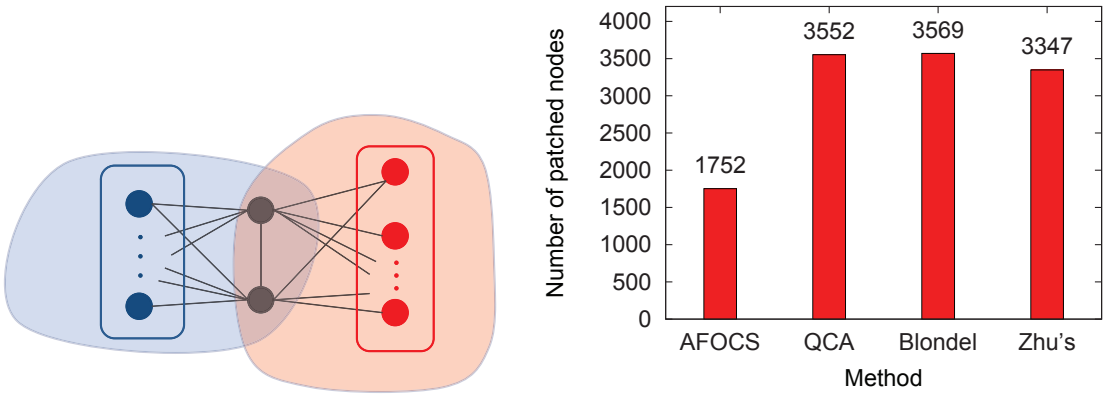
6.2 Containing Worms with Overlapping Communities Detected by AFOCS

We show another application of AFOCS in worm containment problem on OSNs. OSNs are good places for people to socialize online or to stay in touch with friends and colleagues. However, when some of the users are infected with malicious software, such as viruses or worms, OSNs are also fertile grounds for their rapid propagations. Since mobile devices are able to access online social applications nowadays, worms and viruses now can target computers [81] and mobile devices [110].

Recently, community structure-based methods have been proven to be effective solutions to prevent worms from spreading out wider on not only social networks [81][82] but also cellular networks [110]. Due to the high and low frequencies of interactions inside and between communities, worms spread out quicker within a community than between communities. Therefore, an appropriate reaction should first contain worms into only infected communities, and then prevent them from getting outside. This strategy can be accomplished by patching the most influential members who are well-connected not only to members of their community but also to people in other communities.

6.2.1 Setup

In our experiments, we use Facebook network dataset collected in [100]. This data set contains friendship information and wall posts among New Orleans regional network, spanning from Sep 2006 to Jan 2009. The data set contains more than 63.7K nodes (users) connected by more than 1.5 million friendship links. We keep other parameters as well as the "Koobface" worm propagation model the same as [82] for comparison convenience. With the advantages of knowledge overlapping communities, we are able to develop a better and more efficient patching scheme. In particular, we enhance the patching scheme presented in in [82] to take the advantage of the overlap regions: nodes in the boundary of overlapped regions are selected for patching (Figure 6-4A). Alg 16 details the adjusted scheme.



A Influential users selection

B Number of patched nodes

Figure 6-4. *OverCom* patching scheme.

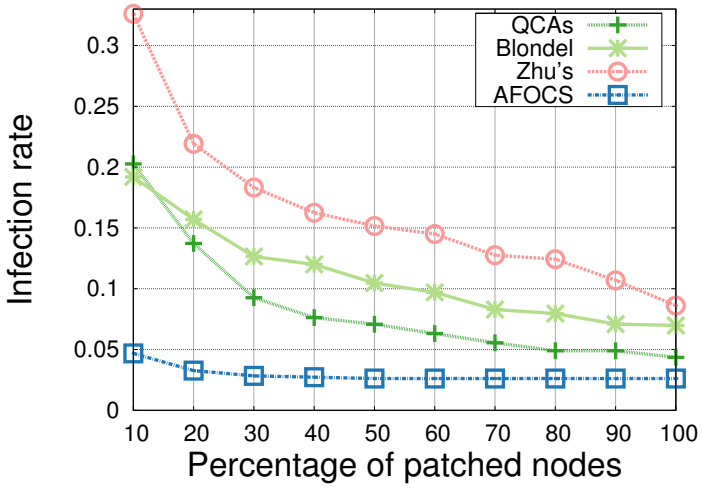
Algorithm 16 *OverCom* Patching Scheme

Input: $G = (V, E)$ and $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ detected by AFOCS
Output: A set of patched nodes IS.

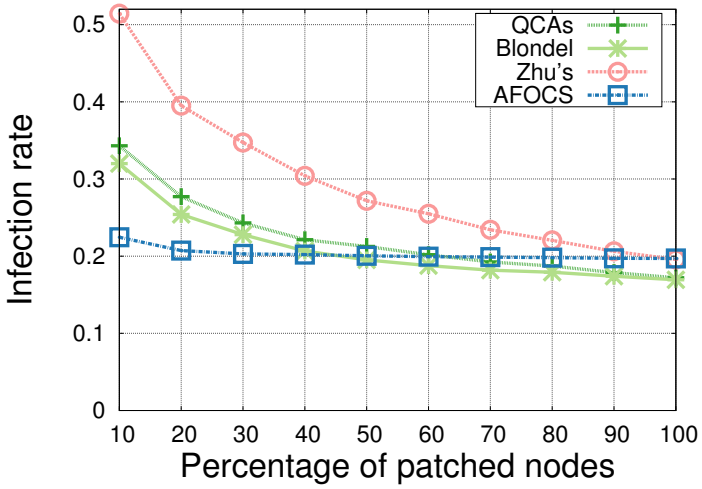
$IS \leftarrow \emptyset$;
for $(C_i, C_j \in \mathcal{C})$ **do**
    if $(C_i \cap C_j \neq \emptyset)$ **then**
        %Choose the neighbors of overlapped nodes as influential ones%
        $IS \leftarrow IS \cup N(u) \quad \forall u \in C_i \cap C_j$;
    end if
end for
%Patch distribution procedure%
for $(u \in IS)$ **do**
    Send patches to u;
    Let u redistribute patches to $w \in IS \backslash N(u)$;
end for

6.2.2 Results

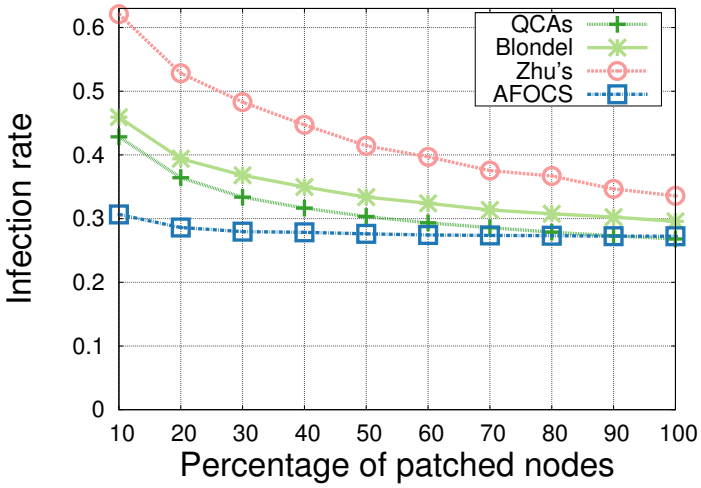
We compare the *OverCom* patching scheme and overlapping communities found by AFOCS to those using disjoint communities proposed by Blondel et al. [6], QCA by Nguyen et al. [81] and Clustering based method suggested by Zhu et al. [110]. The number of patched nodes is shown in Figure 6-4B. Both the number of patched nodes and the infection rates decline remarkably. In particular, the number of nodes to send patch in AFOCS is substantially smaller by half of those required by Blondel, QCA as well as Zhu's methods: only 1725 nodes over 63K nodes in the networks are needed



A $\alpha = 2\%$



B $\alpha = 10\%$



C $\alpha = 20\%$

Figure 6-5. Infection rates between four methods.

to be patched by $OverCom$ patching scheme, while the other schemes require nearly twice ($\geq$3,300 nodes). The reason behind this improvement is due to the nature of our AFOCS framework, the neighbors of the overlapped nodes should not be to far away from the center of each community, thus they can easily redistribute the patches once received.

We next present the achieved infection rates with alarming thresholds (the fraction of infected nodes over all nodes) α = 2%, 10% and 20%, respectively. This threshold alarms the distribution process as soon as the infected rate goes beyond α. The results are reported in Figures 6-5A, 6-5B, 6-5C, respectively. In general, the higher α (i.e., the longer we wait), the more nodes we have to send patches and the higher infection rate. OverCom with AFOCS achieves the lowest infection rates in almost all the experiments and just a little bit lag behind when α = 10%. In particular, when $\alpha = 2\%$, AFOCS helps $OverCom$ to remarkably reduce from 1.6x up to 4.3x the infection rates of QCA, from 2.6x up to 4x the infection rates of Blondel and 3.2x to 7x those of Zhu's method. When α = 10%, AFOCS $+OverCom$ achieves average improved rates of 9% over QCA, 5% over $Blondel$ and 43% over Zhu's methods. As α = 20%, the average improvements are 12%, 23% and 53%, respectively. Due to the nature of the event handling processes, the neighbors of overlapped nodes are not located far away from the rest of their communities. As a result, they can help to distribute patches to more users in the communities, hence help to lower the infection rates of AFOCS. These improvement factors, again, confirm the effectiveness of our proposed method.

CHAPTER 7
STABLE COMMUNITY DETECTION IN ONLINE SOCIAL NETWORKS

A large body of work has been devoted to find general communities (i.e., without the concept of stability) on both directed and undirected networks in the literature [32]. On the contrary, only a very few approaches are suggested to identify stable communities [59][68], especially on directed and weighted networks. The main source of difficulty is due to the inconsistency of community members in a general structure: while they might appear to be in a community at one time, they may not commit to that particular community in a long run. One possible approach, therefore, is to find a consensus of a specific algorithm after multiple runs and use this core as stable communities [59]. However, doing in this way would result in expensive computational cost and time consuming as well as lack of convergence guarantees. In [68], the authors estimate the mutual links between pairs of users and suggest a detection method that optimizes the total mutual connection on the whole network. While the idea of mutual connection is quite interesting, we find that it might not be sufficient because some estimated mutual links are of low magnitudes, and thus, may not reflect the correct concept of stability at the community level.

In general, a stable community is often characterized either by its tight and strong internal relationships represented by the mutual connections among its users [68], or by its internal links who possess a high tendency to remain within the community over a long period of time [23]. In other words, stable communities in the network are commonly characterized by stable connections among their members. Motivated by these observations, we suggest SCD (short for Stable Community Detection), a framework to effectively identify stable communities in directed OSNs that facilitates both of the above intuitions. In a big picture, SCD works by first enriching the input network with the stability estimation of all links in the network, and then discovering communities via stable connections using the lumped Markov chain model. Our approach is

mathematically supported by a key connection between the persistence probability of a community at the stationary distribution and its local topology. One notable advantage of SCD is that it requires only a single iteration, which shall significantly reduce the running time. Furthermore, since our method intrinsically accounts for stability, the discovered communities should be stable as opposed to doing a statistical analysis.

In summary, we suggest an estimation which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD - a framework to identify community structure in directional OSNs with the advantage of community stability. We next explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental mathematical theory to support the SCD framework. To certify the efficiency of our approach, we extensively test SCD on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT and NetHEPT_WC collaboration networks as well as Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

7.1 Basic Notations

We introduce the basic notations representing the underlying social network that we will use throughout this paper.

(Graph notation) Let $G = (V, E, w)$ be a directed and weighted graph representing a social network with V is the set of n network users (or nodes), E is the set of m directed relationships (or edges), and w (or precisely w_{uv}) is the weight function on each edge $(u, v) \in E$ representing the communication frequency between user u and v in the social network. Without loss of generality, we assume that all edge weights are normalized, i.e., $\sum_{(u,v)\in E} w_{uv} = 1$ and $w_{uv} \geq 0$. For each edge $(u, v) \in E$ which $(v, u) \notin E$, we say that the backwards edge (v, u) is missing, we will use the notation

$(v; u)$ and $w_{v;u}$ to denote the mutual link of edge (u, v) if (v, u) should indeed exist in E and its weight, respectively. Furthermore, we will use the notation $st(u, v, t)$ to denote the stability estimate of edge (u, v) at time step (or hop) t. These notations will be described in detail in next section.

(Community notation) Denote by $\mathcal{C} = \{C_1, C_2, ..., C_q\}$ the network community structure, i.e., a collection of q subsets of V satisfying $\cup_{i=1}^{q} C_i = V$ and $C_i \cap C_j = \emptyset \; \forall i, j$. We say that each $C_i \in \mathcal{C}$ and its induced subgraph form a community of G. For a node $u \in V$, let N_u^+, N_u^- and N_u denote the set of outgoing, the set of incoming, and the set of all neighbor nodes adjacent to u, respectively. Furthermore, let k_u^+(or w_u^+), k_u^-(or w_u^-) and k_u(or w_u) be the corresponding cardinalities (or total weights) of these sets. For any $C \subseteq V$, let C^{in} and C^{out} denote the set of links having both endpoints in C and the set of links heading out from C, respectively. In addition, let $m_C = |C^{in}|$ (rsp. $w_C = w(C^{in})$) and $k_C^+ = \sum_{u \in C} k_u^+$ (rsp. $w_C^+ = \sum_{u \in C} w_u^+$). Finally, the terms node-vertex as well as edge-link-connection are used interchangeably.

7.2 Link Stability Estimation

We describe our first step towards the identification of stable communities in the network: the link stability estimation process. Intuitively, a stable community is often characterized either by its tight and strong internal relationships represented by the mutual connections among its users [68], or by its internal links who possess a high tendency to remain within the community over a long period of time [23]. In other words, stable communities in the network are commonly characterized by stable connections among their members. Motivated by these observations, in this section, we suggest a procedure for estimating the stability of each link in the network that facilitates both of the above intuitions. Our estimation procedure consists of two stages: In the early stage, the reciprocity of each link in the network is first predicted, and based on that, its stability is consequently evaluated in the later stage.

7.2.1 Link reciprocity prediction

When dealing with large scale OSNs, it is possible that some backwards edges between individuals are missing. This lack of information may due to the imperfect data collection process, or because these backwards edges are not yet reflected in the underlying network but should due to the strong relationships between local network users. For instance, Leskovec et al. [65] observe that friends of friends in social networks tend to be friend of each other in the near future, i.e., there should be dual connections between friends of friends with high chance even if they are not yet friend of each other. Therefore, predicting the existence of these backwards edges will allow a more complete and comprehensive detection of stable communities by increasing the internal density of strongly connected components, which are potential candidates for network communities.

Link reciprocity prediction problem is a well-studied field and many methods are proposed in the literature [69][3][30]. In this paper, we utilize a method called "friends-measure" suggested in [30]. The intuition behind this measure is that when looking at two users in the social network, one can assume that the more connections their neighbors have with each other, the higher the chance the two users are actually connected. Originally, this friends-measure between two users u and v is formulated as:

$$\text{friends-measure}(u, v) = \sum_{x \in N_u} \sum_{y \in N_v} \delta(x, y)$$

where $\delta(x, y) = 1$ if either $x = y$ or $(x, y) \in E$ or $(y, x) \in E$. This measure has been extensively verified among other topological features and has been shown to be a promising one in comparison with other metrics [30]. However, in the case of directed networks, there are possibilities that different link topologies can share a common friends-measure value. Therefore, we need to modify the above formula so that it reflects the true relationship between the network users, and furthermore copes with edge weights in the network.

In order to better handle directed and weighted graphs, we will attempt to predict the existence of backwards edges of unidirectional links. For example, if $(u, v) \in E$ and $(v, u) \notin E$, we will try to find the possibility whether we should enrich the network by inserting (v, u) into E. To this end, we first relax the direction of the edge between u and v, and next compute the likelihood that a backwards edge should exist between u and v by using the modified formula

$$\Delta(u, v) = \frac{\sum_{x \in N_v} \sum_{y \in N_u} \tau(x, y)}{W_v W_u} \tag{7–1}$$

Where $\tau(x, y) = w_{vx} w_{xy} w_{yu}$ is the total possibility of the backwards path starting from v, passing through neighbor nodes x and y, and ending at u. When the network is unweighted $W_u = d_u$, $W_v = d_v$ and thus, $\Delta(u, v)$ counts the (normalized) number of paths of lengths two and three joining two users u and v, which intuitively agrees with the aforementioned friends-measure formula. By Proposition 7.1, we show that $\Delta(u, v)$ is indeed the generalization of weighted friend-measure(u, v) and depends only on the nodes' topology. Hence, $\Delta(u, v)$ can be regarded as the estimated probability that the backwards connection $< v, u >$ indeed exists, i.e., we set $w_{<vu>} = \Delta(u, v)$.

Proposition 7.1. *For any* $(u, v) \in E$ *which* $(v, u) \notin E$*,* $0 \leq \Delta(u, v) \leq 1$.

Proof. We first prove this for unweighted graphs. The proof for weighted graphs can be extended straightforwardly. It is obvious that $0 \leq \Delta(u, v)$. Now we show $\Delta(u, v) \leq 1$. For any x, y such that $\delta(x, y) = 1$, if $x = y$, they can make just one connection counted towards the summation. Otherwise, they can make at most d_u (or d_v) dual-connections at each vertex. Taking these facts into account, we have $\Delta(u, v) = \sum_{x \in N_v} \sum_{y \in N_u} \delta(x, y) \leq d_v d_u$. Thus, the inequalities follows. The left equality holds when there are no connections from u to v and vice versa. The right equality holds when every path of length 2 from u to v (or from v to u) are contained in the corresponding path of length 3. □

Proposition 7.2. *Let n_0 be the number of unidirectional links in the input network. The time complexity for estimating the mutual connections for these links is $O(n_0 M)$.*

Proof. The total time required for estimating the possibility for a backward connection at an edge (u, v) is $d_u + d_v + \min \sum_{x \in N_v^+} \sum_{y \in N_y^-} \{d_x^+, d_y^-\}$. Thus, for all n_0 links, the total time complexity is upper bounded by $n_0(2M) + n_0 M = O(n_0 M)$. □

7.2.2 Link stability estimation

After the reciprocity of each link in the network has been estimated, the input network is now enriched with more information of the backwards edges. While the presence of these dual edges is helpful in characterizing the mutual relationships between pairs of network users, it might not be sufficient to evaluate the stability of all network connections as some of the backwards edges may be of low magnitudes, and thus, may not be able to hint the stability of the connection. Therefore, we need to further estimate the stability of a network link given its predicted reciprocity. In order to do so, we define the stability of an edge $(u, v) \in E$ at t time steps (or t hops) as follow

$$st(u, v, t) = \sum_{|P|=t} w(P)$$

where P is a path going from v to u (v and u are excluded) of length $|P| = t$, and $w(P) = \prod_{(a,b)\in P} w_{ab}$ is the total weight of path P. Finally, we define the stability $st(u, v)$ of a link $(u, v) \in E$ as the total stability of up to T_0 time steps, where T_0 is a predefined parameter (or the upper bound on the number of hops)

$$st(u, v) = \sum_{t=1}^{T_0} st(u, v, t). \tag{7–2}$$

The intuition behind our stability function $st(u, v)$ is as follow: since stable communities are commonly recognized by a high density of stable edges, it is reasonable to expect that such edges form a cycles. In the senses of directed and weighted networks, the stronger the strength of cycles an edge (u, v) is on, the more stable it is believed to be.

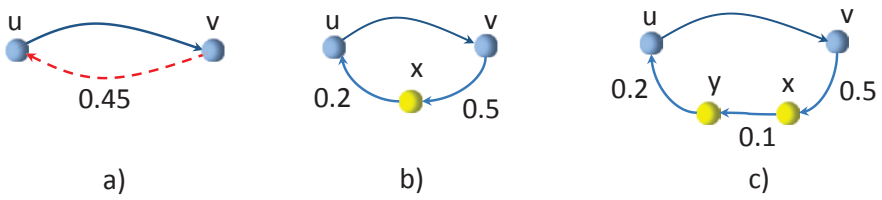


Figure 7-1. Illustrations of stability function.

On the contrary, edges that connecting or joining between communities shall hardly be part of many cycles, and eventually result in low stability. Figure 7-1 illustrates the stability estimates for link (u, v) at 0, 1 and 2 hops: a) $st(u, v, 0) = 0.45 = w_{<vu>}$, b) $st(u, v, 1) = 0.5 \times 0.2 = 0.1$, c) $st(u, v, 2) = 0.5 \times 0.1 \times 0.2 = 0.05$.

As a local measure, our suggested stability function has the following advantages (1) it puts more focus on the existence of the mutual link of any link (u, v) by reserving the original strength of the backwards edge $< v, u >$. This intuitively agrees with the findings that stable clusters are usually made of bidirectional links in [68]. Moreover, our formula further takes into account the strength of cycles containing the current link; (2) the more time (or, number of hops) we allow, the more stability a link would be. Nevertheless, links that really belong to a stable community are more likely to have strong stability whereas those connecting communities are of low stability. These advantages support the intuitions of stable communities that we discussed above. The performance of our stability estimation is evaluated in more detail in section 7.4.

In summary, our link stability estimation first predicts the potential of the dual link of any link $(u, v) \in E$ such that $(v, u) \notin E$ by using the modified measure in equation (7–1). Next, it evaluates the stability of the every link in the given network enriched from the first stage by using equation (7–2), and utilizes these stability values as new weights for links in the network. This resulting network will be consequently passed as the input network to our main process: the identification of stable communities.

7.3 Stable Community Detection

In this section, we present our main contribution: the stable community identification process. Given the input network enriched with link stability information, we discover the stable communities by exploring an important connection between the persistence probability of each community and its local network topology. In the following paragraphs, we first review the concept of Lumped Markov chain [50][89], and then establish our key connection between this Markov chain and the local network topology. Finally, we describe in detail our last but most important process: stable community detection.

7.3.1 Lumped Markov chain

A Markov chain [96] is a mathematical system representing transitions from one system's state to another, between a finite number of predefined states. In terms of social networks, a state can be either a user (a node in the graph) or a group of tightly connected users (a community) in the networks, whereas transitions can be regarded as the user-to-user or group-to-group communication tendencies. An n-state Markov chain corresponding to an n-node network is commonly represented by the transition $\pi_{t+1} = \pi_t P$, where $\pi_t = (\pi_{1,t}, \pi_{2,t}, \ldots, \pi_{n,t})$ with $\pi_{u,t}$ is the probability of being at node u at time t, and $P = (p_{uv})$ is the transition matrix. In particular, this n-state Markov chain can be associated to input network by letting the probability of transiting from a node u to a neighbor node v as

$$p_{uv} = \frac{w_{uv}}{\sum_j w_{uj}} = \frac{w_{uv}}{w_u^+}.$$

Basically, p_{uv} is the probability of a random walker jumps from node u to node v given the network topology. A Markov chain is said to be at its stationary state distribution π if π satisfies the equation $\pi = \pi P$. As shown in [43], when the network is originally connected P would be irreducible, and thus, the equation $\pi = \pi P$ has a unique solution which is strictly positive ($\pi_u > 0 \; \forall u \in V$) which corresponds to the stationary Markov chain state distribution. When the network is undirected, π can be exactly computed as $\pi = \frac{1}{2W_0}(w_1, w_2, \ldots, w_n)$ with W_0 is the total edge weights. However, we do not have an

exact form for the stationary distribution π in general for directed network, and thus, π has to be computed numerically.

As our ultimate goal is to detect the stable network community structure, we sought to find a good partitioning of V where each partition will remain wealthy over time. In the light of Markovian chain method, this corresponds to finding a collection of communities $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ where a random walker would spend most of the time walking inside a community and less time wandering among communities. By defining this partition $\mathcal{C}$ of q communities, we introduce a so called q-state meta-network where each community in the network becomes a meta-state. However, at this aggregate level, a in general dynamics Markovian description of a random walker walking among communities is not possible because the Markovian property may not be well-preserved [50]. Nevertheless, this q-state community-to-community transition can still be defined using the lumped Markov chain, which correctly describes the random walker at this scale given the stochastic process is started at the stationary distribution π [43]. This lumped Markov chain is defined via the $q \times q$ matrix as U in [89]

$$U = [diag(\pi H)]^{-1} H^{\mathsf{T}} diag(\pi) P H$$

where H is a $n \times q$ binary matrix representing the partitioning $\mathcal{C}$.

One of the notable advantages of the lumped Markov chain $\Pi_{t+1} = \Pi_t U$ defined on U is that it shares the same stationary distribution with the original Markov chain, i.e., the new stationary distribution defined by $\Pi = \pi H$ satisfies the equation $\Pi = \Pi U$. Moreover, the difference between $\Pi_{t+1} = \Pi_t U$, starting at $\Pi_0 = \pi U$, and the original $\pi_t H$ tends exponentially to zero if the two chains are regular. These advantages make the community-based lumped Markov chain defined by $\Pi_t U$ a very good approximation of the original n-node network. We stress that the ability of the lumped Markov chain to describe the random walk dynamics only at stationary is not a limitation for the detection of stable communities. Indeed, this stationary requirement evaluates the random walk

dynamics of all nodes at their stable states, and hence perfectly supports the concept of stable communities.

In terms of interpretation, each entry u_{cd} of U denotes the chance that a random walker, at time t, wanders from community c to another community d in time $t+1$. As a result, the diagonal elements u_{CC}'s (or u_C's in short) of U indicate the persistence probabilities that a random walker just walking within a particular community C. Of course, large values of u_C's are expected for meaningful communities. It is also shown in [89] that in directed and weighted graphs, u_C can be computed as

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} \tag{7–3}$$

Note that $\sum_{i,j\in C} \pi_i p_{ij}$ is the fraction of time a random walker spends on the links inside a community C. Hence, u_C is indeed the ratio between the amount of time a random walker spends on links and that it spends on nodes in C. In undirected networks, one can verify that

$$u_C = \frac{\sum_{i,j\in C} \pi_i w_{ij}}{\sum_{i\in C} w_i} = \frac{2w_C}{2w_C + w(C^{out})}.$$

7.3.2 Connection to the network topology

At this stage, one might try to optimize u_C for all communities $C \in \mathcal{C}$ in order to maximize theie persistence probabilities. However, doing in this way requires solving for the stationary distribution π_i's (as in equation (7–3)) which may be extremely costly, especially in large scale directed networks. So, how can we effectively optimize the persistence probability u_C for each community without solving for that costly exact stationary distribution? As an answer for this challenging question, we present in Proposition 7.3 a connection between the persistence probability of a community C and its local topology. In particular, we show that the minimum value of u_C can be represented by quantities that only involve C's local topology. Therefore, optimizing u_C

can be shifted as the optimization of these local components, which are inexpensive and easy to derive.

Proposition 7.3. *For any community $C \in \mathcal{C}$, at the stationary distribution π, we have the following inequality*

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} \geq \frac{w_C}{w_C^+}.$$

Proof. It is easy to see that

$$u_C = \frac{\sum_{i,j\in C} \pi_i p_{ij}}{\sum_{i\in C} \pi_i} = \frac{\sum_{i\in C} \pi_i \frac{w_{i,C}}{w_i^+}}{\sum_{i\in C} \pi_i}.$$

where $w_{i,C} = \sum_{j\in C} w_{ij}$. Next, we rewrite $\sum_{i\in C} \pi_i$ in the form $\sum_{i\in C} \pi_i = \pi^\top e_C$ where $e_C = (e_i)_{N\times 1}$ and $e_i = 1$ if $i \in C$ and 0 otherwise. Since π is the stationary distribution of the Markov chain, we have $\pi = \pi P$. Thus

$$\pi^\top e_C = \pi^\top P e_C = \sum_{i\in C} \pi_i \Big(\sum_{j:(i,j)\in E} \frac{1}{w_i^+} \Big)$$

Now we have,

$$\begin{aligned} \sum_{i\in C} \pi_i \times w_C &= \sum_{i\in C} \pi_i \Big(\sum_{j:(i,j)\in E} \frac{1}{w_i^+} \Big) w_C \\ &\leq \sum_{i\in C} \pi_i \frac{w_{i,C}}{w_i^+} \Big(\sum_{t\in C} w_t^+ \Big) = \sum_{i,j\in C} \pi_i \frac{w_{i,C}}{w_i^+} \times w_C^+ \end{aligned}$$

Hence, the conclusion follows. The quality holds when all π_i equals to each other and $w_C = w_C^+$. This happens when C is a full dually connected clique and is disconnected from the rest of the network. □

7.3.3 Detecting communities

7.3.3.1 Formulation

Proposition 7.3 discussed in the above paragraph establishes the connection between the persistence probability of a random walker staying within a community C and the local network topology. As a result, if we can maximize the later quantity, we can provide some insurance to the desired optimization with high confidence. Taking into

account this intuition, we propose Stable Community Detection (SCD) as an optimization problem defined as follow: Given a directed, weighted network $G = (V, E, w)$, find a community structure $\mathcal{C} = \{C_1, C_2, ..., C_q\}$ such that the overall total persistence probability is maximized:

$$\max \mathcal{R} = \sum_{C \in \mathcal{C}} \frac{w_C}{w_C^+}$$

subject to

$$C_i \cap C_j = \emptyset \qquad \forall i, j \in \{1, 2, ..., q\}$$

$$\bigcup_{i=1}^{q} C_i = V$$

Note that in our SCD formulation, the number of communities q will be determined by optimizing the objective function $\mathcal{R}$ and is not an input parameter. Indeed, optimizing $\mathcal{R}$ provides us q a very good estimate for the actual number of communities, as we will show in section 7.4.

7.3.3.2 Resolution limit analysis

Perhaps one of the most important properties a metric suggested for identifying community structure should satisfy is the ability of overcoming the resolution limit [35], i.e., the metric should be able to detect network communities even at different scaling levels. In this subsection, we analyze the resistance to resolution limit of our proposed function $\mathcal{R}$ by looking particularly at the condition in which two communities should be merged together. In what following, we simplify the situation by considering undirected networks.

Let us consider two communities C_1 and C_2. Let m_{12} be the number of edges connecting C_1 and C_2. In order to merge C_1 and C_2 into a bigger community, m_{12} should satisfy:

$$\frac{m_{C_1}}{d_{C_1}^+} + \frac{m_{C_2}}{d_{C_2}^+} \leq \frac{m_{C_1} + m_{C_2} + m_{12}}{d_{C_1}^+ + d_{C_2}^+}$$

The above condition is equivalent to:

$$\frac{m_{C_1}}{d^+_{C_2}} d^+_{C_1} + \frac{m_{C_2}}{d^+_{C_1}} d^+_{C_2} \leq m_{12}$$

which in turn implies $2\sqrt{m_{C_1} m_{c_2}} \leq m_{12}$. Without loss of generality, we can assume that $m_{C_1} \leq m_{C_2}$, thus $2m_{C_1} \leq m_{12}$. This violates the condition of even a weak community. Moreover, this inequality implies the sufficient condition to merge two adjacent communities depends on the local structure of two communities only, regardless of the rest of the network. This observation indicates that our proposed metric $\mathcal{R}$ is strongly against the resolution limit.

7.3.3.3 Connection to stability estimation

We next verify the following properties of network communities identified by optimizing our suggested metric $\mathcal{R}$: (1) links within a communities are of high stability and (2) links connecting communities are of low stability values. These two observations are shown in Propositon 7.4.

Proposition 7.4. *Let $\mathcal{C} = \{C_1, C_2, \ldots, C_k\}$ be a community structure detected by optimizing $\mathcal{R}$, links within each C_i are of strong stability and those connecting communities are of weak stability values.*

Proof. For any node $p \in V$ and subset $A \subseteq V$, let $w_{p,A}$ be the total weight of all links that p has towards A and vice versa. By this definition, we obtain $w_p = w_{p,A} + w_{p,V\setminus A}$. For any community $C \in \mathcal{C}$, $s \in C$ and $p \notin C$, since p is not a member of C, we have

$$\frac{w_C}{w^+_C} > \frac{w_C + w_{p,C}}{w^+_C + w_p} = \frac{w_C + w_{p,C}}{w^+_C + w_{p,C} + w_{p,V\setminus C}},$$

because otherwise joining p to C will give a better value of $\mathcal{R}$. This equality equals

$$\frac{w_{p,C}}{w_p} < \frac{w_C}{w^+_C},$$

which in turn implies that the stability contribution of links joining p to C are insignificant in comparison to C as a whole.

Similarly, for any node $s \in C$, we have

$$\frac{w_C}{w_C^+} > \frac{w_C - w_{p,C}}{w_C^+ - w_p} = \frac{w_C - w_{p,C}}{w_C^+ - w_{p,C} - w_{p,V\setminus C}},$$

because otherwise excluding s from C will give a better $\mathcal{R}$. This inequality equals to

$$\frac{w_{s,C}}{w_s} > \frac{w_C}{w_C^+},$$

which in turn implies that the stability contribution of internal links of C are significant in comparison to C as a whole. □

7.3.3.4 A greedy algorithm for SCD problem

Analyzing the theoretical hardness of the SCD problem is an aspect beyond the scope of this paper. In fact, the NP-hardness of the SCD problem can be shown by a similar reduction to MODULARITY as in [8] (see also [101] and [36] for a comprehensive survey on similar graph clustering problems). Given its NP-hardness, a heuristic approach that can provide a good solution in a timely manner is therefore more desirable. In this section, we describe a greedy algorithm for the SCD problem consisting of community growing, strengthening and refinement phases described as follow.

Growing phase. This phase is responsible for discovering raw communities in the input network. Initially, all nodes are unassigned and do not belong to any community. Next, a random node is selected as the first member (or the seed) of a new community C, and consequently, new members who help to maximize C's persistence probability are gradually admitted into C. When there is no more node that can improve this objective of the current community, another new community is formed and the whole process is then cycled in the very same manner on this newly formed community.

Strengthening phase. We further rearrange nodes into more appropriate communities. Since new members are admitted into a community C in a random order, C's objective value could be further improve with the absence of some of it members as they can be

Algorithm 17 SCD Algorithm

Input: A directed weighted graph $G = (V, E, w)$
Output: Community structure $\mathcal{C}$

Growing Phase:
$\mathcal{C} \leftarrow \emptyset$
$A \leftarrow V$
while $\exists$ unassigned node $u \in A$ **do**
  $C \leftarrow \{u\}$
  $A \leftarrow A \setminus \{u\}$
  while $\exists\ v \in A$ such that $u_{C\cup\{v\}} > u_C$ **do**
    $v \leftarrow \arg\max_{v\in A}\{u_{C\cup\{v\}}\}$
    $C \leftarrow C \cup \{v\}$
    $A \leftarrow A \setminus \{v\}$
  end while
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C\}$
end while

Strengthening Phase:
for $C \in \mathcal{C}$ **do**
  while $\exists u \in C$ such that $u_C < u_{C\setminus\{u\}}$ **do**
    $C \leftarrow C \setminus \{u\}$
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{u\}$
  end while
end for

Refining Phase:
while $\exists C_1, C_2$ such that $u_{C_1\cup C_2} > u_{C_1} + u_{C_2}$ **do**
  $(C_1, C_2) \leftarrow \arg\max_{C_1,C_2\in\mathcal{C}}\{u_{C_1\cup C_2} - u_{C_1} - u_{C_2}\}$
  $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C_1, C_2\}) \cup \{C_1 \cup C_2\}$
end while
Return $\mathcal{C}$

obstacles for the total stability. This requires the reevaluation of all C's members as a result. Therefore, in this phase, we exclude any node which reduces the persistence probability of a community and let them be singleton communities. The removal of such nodes creates more cohesive communities, i.e., communities with higher internal stability.

Refining phase. In the last phase, the global stability of the whole network is reevaluated. In particular, this last refinement phase looks at the merging of

two adjacent communities in order to improve the overall objective function. If two communities have a great number of mutual connections between them, it is thus more stable to merge them into one community. The final algorithm, which we call SCD algorithm, is presented in Alg. 17.

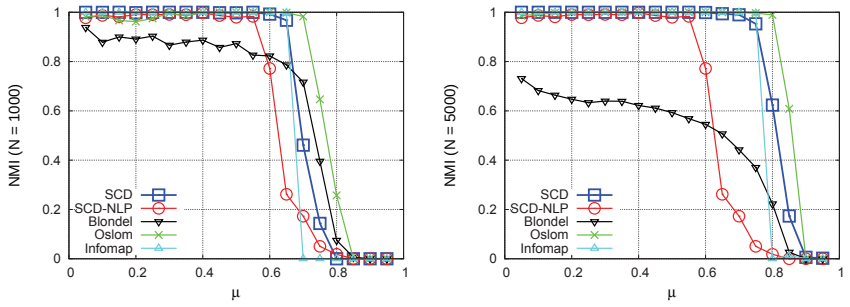
7.4 Experimental Results

In this section, we present our results on the discovery of network communities on both synthesized networks with known groundtruths and real-world social traces including NetHEPT and NetHEPT_WC collaboration and Facebook networks. We evaluate the following aspects of our proposed SCD framework (1) the effectiveness of our link stability estimation process, (2) the ability of identifying the general network community structure without the concept of community stability, i.e., how similar our detected communities are in comparison with the groundtruths, and (3) the ability of identifying stable communities in reference to the consensus of other state-of-the-art methods, including Blondel's [6], Infomap [93] and OSLOM [61] methods, after their multiple executions.

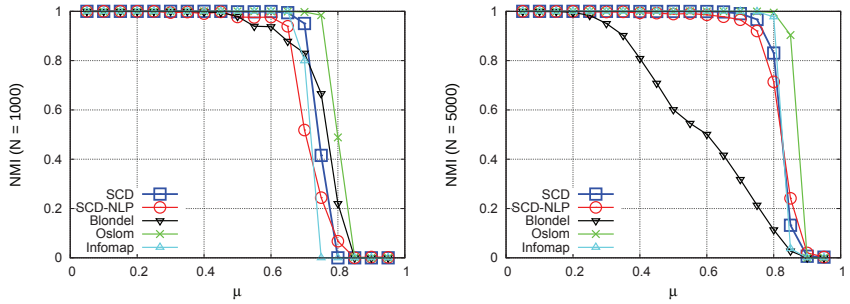
7.4.1 Datasets

(Synthesized networks) Of course, the best way to evaluate our approaches is to validate them on real-world networks with known community structures. Unfortunately, we often do not know that structures beforehand, or such structures cannot be easily mined from the network topologies. Although synthesized networks might not reflect all the statistical properties of real ones, they can provide us the known groundtruths via planted communities and the ability to vary other network parameters such as sizes, densities and overlapping levels, etc. Testing community detection methods on generated data has also becomes a usual practice that is widely accepted in the field [55].

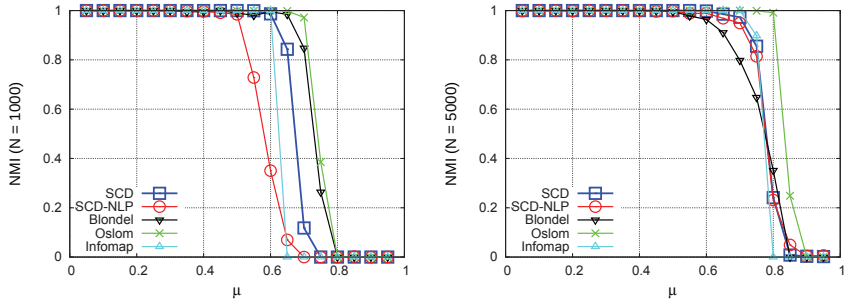
We use the well-known LFR benchmark [55] to generate 190 weighted and directed testbeds. Generated data follow power-law degree distribution and contain embedded



A Networks with $minC$, $maxC$ unconstrained.



B Networks with $minC = 25$, $maxC = 50$ (small-size).



C Networks with $minC = 50$, $maxC = 100$ (big-size).

Figure 7-2. Results on synthesized networks with different community criteria.

communities of varying sizes that capture characteristics of real-world networks. Parameters are: the number of nodes $N = 1000$ and 5000, the mixing parameter $\mu = [0.1...1]$ controlling the overall sharpness of the community structure, the minimum ($minC$) and maximum ($maxC$) of community sizes are set to (25, 50) for small-size and (50, 100) for big-size communities as in the standard settings. Each test is averaged over 100 runs for consistency.

(NetHEPT and NetHEPT_WC) The NetHEPT traces are widely-used datasets for testing social-aware detection methods [11][12]. These traces contain information, mostly the academic collaboration from arXiv's "High Energy Physics - Theory" section where nodes stand for authors and links represent coauthorships. In their deliverable, the NetHEPT networks contain 15233 nodes and 31398 links, and weights on edges are assigned by either uniformly at random (for NetHEPT data) or by weighted cascade (for NetHEPT_WC data) where $w_{uv} = 1/d_{in}(v)$ with $d_{in}(v)$ is the indegree of a node v.

(Facebook) This dataset contains friendship information among New Orleans regional network on Facebook, spanning from September 2006 to January 2009 [100]. The data contains more than 63K nodes (users) connected by more than 1.5 million friendship links with an average node degree of 23.5. In our experiments, the weight for each link between users u and v is proportional to the communication frequency between them, normalized on the whole network.

7.4.2 Metric

To measure the quality of the detected communities in comparison with the embedded groundtruths, we evaluate Generalized Normalized Mutual Information (NMI) [55]. Basically, the NMI(U,V) value of two structures U and V is 1 if U and V are identical and is 0 if they are totally separated. This is the most important metric for a community detection algorithm because it indicates how good the algorithm is in comparison with the planned communities. Higher NMI values are expected for a better community detection algorithm.

7.4.3 Effect of link stability estimation

We first evaluate the effect of our link stability estimation on the detection of network communities by comparing NMI values of SCD and its version with No Link stability Prediction (SCD-NLP). Due to space limit, results of SCD and SCD-NLP are also reported in Figure 7-2, where those on general community structure detection are also presented.

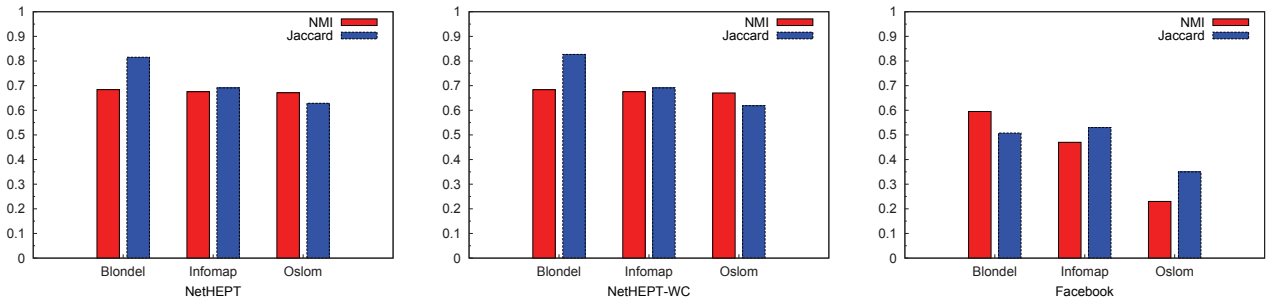


Figure 7-3. Performance of SCD in detecting stable communities on real social traces.

In general, SCD-NLP performs very competitively even without being preprocessed: on synthesized networks with no community size constraint (Figure 7-2A), its discovered communities are almost of perfect similarity to the embedded ones (NMI values approximately 1) on $\mu = [0...0.5]$ whereas the quality drops down quickly when μ is above 0.5. We note that this drop of detection quality is controversial and does not necessary imply a bad performance since networks with $\mu > 0.5$ is consider very stochastic, and thus, may not contain a clear community structure. Nevertheless, with the help of the stability estimation, the performance is now boosted up significantly on SCD as the detection qualities are very high even for $\mu > 0.65$ ($N = 1000$) and $\mu > 0.75$ ($N = 5000$), and only drop down when the networks are extremely stochastic ($\mu > 0.8$).

We next take a look at the cases where networks are constrained with small-sized (Figure 7-2B) and big-sized communities (Figure 7-2C). We observe that, when community sizes are constrained, SCD-NLP performs much better than before and even overcome its prior limit $\mu = 0.5$. In particular, the performance of SCD-NLP closely approaches that of SCD, especially in large networks ($N = 5000$). However, SCD-NLP appears to be sensitive to big-size communities in small networks as its quality drops down quickly in Figure 7-2C (left), and seems to favor small-size communities as its plots tend to tangle with those of SCD (Figure 7-2B). SCD detection quality, thanks to the stability estimation, stays wealthy in all test cases.

In summary, these results indicate that (1) without the stability estimation process, our suggested metric $\mathcal{R}$ appears to be a very good one to detect community structure in general directed and weighted networks, and (2) when the community size is constrained, link stability estimation has a little effect on the community detection quality. However, in real-world social network settings where community sizes are typically unknown, and therefore unconstrained, the stability estimation has a significant effect on the detection of network communities. These experiments also confirm the efficacy of our proposed stability estimation procedure.

7.4.4 General community structure detection

We next investigate on SCD's ability to identify general network community structure, i.e., without community stability, in comparison with the aforementioned state-of-the-art detection methods. Results are reported in Figure 7-2.

In general, the performance of our SCD frameworks on synthesized networks appears to be better than those of Blondel and Infomap methods, and only lags behind Oslom's when the networks are heavily stochastic. When the community size is unconstrained, the detection quality of SCD and other methods, except for Blondel's, retain at nearly perfect on $\mu = [0...0.65]$ ($N = 1000$) and $\mu = [0...0.8]$ ($N = 5000$) and then all degrade quickly. Among the three methods, Infomap's performance appears to be sensitive to some certain mixing threshold μ as it NMI values tend to drop directly to 0, whereas Oslom and ours tend to drop down slower. On average, the NMI values of SCD are about 8% and 3% better than those of Blondel and Informap methods, and are about 2% lag behind those of Oslom method. Blondel's method, on the other hand, does not attain a good performance through due to low NMI values even at a low range of mixing value μ. An possible explanation for this behavior of Blondel's method is due to the effect of resolution limit, as we shall discuss below.

When the embedded communities are constrained with small and large community sizes, we observe the nearly same behavior of SCD, Oslom and Infomap methods as

depicted in Figures 7-2B and 7-2C. Blondel's method gets a significant improvement in these cases where its performance is closely related to the others. As we discussed above, one possible reason for the bad behavior of Blondel's method is due to the resolution limit of modularity objective function [35]. As the community size is unconstrained, this resolution limit can mislead Blondel method to merge some communities whose are of small sizes in comparison to the rest of the network, thus results in the low NMI values. On the other hand, this resolution limit does not take effect when size constraints are imposed and thus the significant improvement. Our SCD framework, as shown in section 7.3.3.2, can withstand this scaling limit as its obtains highly competitively results. Moreover, the difference between our SCD and other methods are insignificant on average which indicates that all methods are able to detect network communities with high quality. This is not a surprising result since Blondel, Oslom and Informap are currently state-of-the-art methods but a great motivation and award for our SCD framework.

7.4.5 Results on stable community detection

In order to compare our results to the consensus of other detection methods, we will adopt a strategy recently proposed in [59]. In particular, given a specific community detection method $\mathcal{A}$, its consensus (or stable) communities can be determined by: (i) execute $\mathcal{A}$ on G n_p times to have n_p partitions (ii) find the matrix $D = (D_{ij})$ where D_{ij} is the probability which vertices i and j of G are assigned to the same cluster among n_p partitions (iii) all D_{ij}'s that are below a threshold τ will be disregarded (iv) Apply $\mathcal{A}$ on D n_p times, so to create n_p partitions and (v) if all partitions are equal, stop (the result matrix would be block diagonal). Otherwise go to step (ii). As suggested in [59], the resulted communities are ideal candidates for stable structures as members commit to their communities. We also compute the Jaccard index $J(U, V) = \frac{|A \cap B|}{|A \cup B|}$ to better evaluate the quality of the detected stable communities. Results are represented in Figure 7-3.

As illustrated by the subfigures, even a single run of SCD is able to obtain very high NMI scores and Jaccard indicies in comparison with the consensus of other methods after multiple runs. In particular, community structures discovered by SCD on NetHEPT and NetHEPT_WC obtain nearly 70% similarity in comparison with Blondel, Infomap and Oslom methods, meanwhile the Jaccard indicies indicate that, in average, almost 66% number of nodes are found in common between SCD and the core structure of other competitors. This show that communities discovered by SCD are indeed highly overlap with core community structures identified by other detection methods, which in turns implies that those clusters found by SCD are stable with high confidence. Surprising, in both NetHEPT and NetHEPT_WC networks, we observe the high similarity among the consensus of Blondel, Infomap and Oslom methods even with difference in edge weight distribution. This observation indicate those identified communities by SCD are, in fact, stable in these networks.

Even in Facebook, a large network with real social interactions, the similarity between consensus communities discovered by other methods and by SCD are still of high similarity with nearly 60%, 50% similarity to those found by Blondel and Infomap methods with over 50% overlap in the stable partitions as indicated by the Jaccard indices. The achieved NMI values in comparison with Oslom method are relatively low as their core communities do not appear to highly overlap (Jaccard index of only 35%). We note that this low similarity does not indicate the unstability community structure of our SCD framework since communities detected by Oslom can be overlapped with each other, while SCD works towards disjoint community structure. Nevertheless, as just a single run, the above competitively results in reference to other state-of-the-art methods confirm the efficay and quality of our method in detecting stable network communities in OSNs.

7.5 Conclusion

In this work, we investigate community structures in directed OSNs with more focus on community stability. As an effort towards the understanding of stable communities, we suggest an estimation procedure which provides helpful insights into the stability of links in the input network. Based on that, we propose SCD, a framework to identify community structure in directed OSNs with the advantage of community stability. We explore an essential connection between the persistence probability of a community at the stationary distribution and its local topology, which is the fundamental point to back our SCD framework. Finally, we certify the efficiency of our approach on both synthesized datasets with embedded communities and real-world social traces, including NetHEPT collaboration and Facebook social networks, in reference to the consensus of other state-of-the-art detection methods. Highly competitive empirical results confirm the quality and efficiency of SCD on identifying stable communities in OSNs.

CHAPTER 8
ASSESSING NETWORK COMMUNITY STRUCTURE VULNERABILITY

8.1 Introduction

As a first study on assessing the vulneraibility of the network community structure, in this paper, we take the first step on understanding how the failures of crucial nodes in the network will affect its community structure. Particularly, we are interested in identifying network nodes whose removals trigger a significant restruction of the current community structure. Formally, given the input network and a positive number k, we introduce the Community Structure Vulnerability (CSV) which aims to find out a set S of k nodes whose removal maximally transforms the current network community structure to a totally different one, i.e., the new community structure resulted from the removal of S is of least similarity to the original one, evaluated via the Normalized Mutual Information [20] measure.

Knowledge about this crucial vulnerability of network community structure is of considerable usage, especially for social-aware methods in mobile ad-hoc and online social networks (OSNs). To give a sense of its effects, consider message forwarding in DTNs. Since social-based forwarding strategies in DTNs rely on the highest ranked nodes in each community to forward the message [47][80], the knowledge of this vulnerability can help to either design routing algorithms that do not overload those crucial devices, if they are those highly ranked ones in a community, or to design an effective backup plan when some of them may fail at the same time. In worm containment application in OSNs [82][110], this knowledge can provide helpful insights into the protection of those sensitive nodes, if they are indeed high influential users, once worms spread out in the network. As a result, the identification of nodes whose removal triggers a massive restruction of the community structure is extremely important for the network's regular operation. However, under a minor structural change when a node is excluded from a community, this particular community can either stay intact if the

removed node is less important, or can be broken down into smaller subcommunities which can further be merged to other communities if the current node is of great important to the community. This unpredictable transformation of network communities together with their large scales in reality make the assessment of community structure vulnerability a fundamental yet challenging problem.

8.2 Problem Definition

In this section, we first define the graph notations that will be used thoroughly in this paper. We then describe Normalized Mutual Information (NMI) [20], a concept in Information Theory, as a metric to assess the difference between community structures before and after the removal of important nodes. Finally, we formally define the Community Structure Vulnerability problem - our main focus in this paper.

(Notations) Let $G = (V, E)$ be an undirected unweighted graph representing a network where V is the set of $|V| = N$ nodes (e.g., users), and E is the set of $|E| = M$ links. For any node $u \in V$ and a set $C \subseteq V$, let $N(u)$, d_u and d_u^C be the set of all neighbors of u, its degree in G and its degree in C, respectively. Furthermore, let $n_C = |C|$ be the number of nodes and m_C be the number of internal edges in C.

(Community structure) Denote by $\mathcal{A}$ the specific community detection algorithm that will be applied on G, and by $X = \{X_1, X_2, ..., X_{c_X}\}$, $Y = \{Y_1, Y_2, ..., Y_{c_Y}\}$ the two (possibly overlapped) community structures of c_X and c_Y communities detected by $\mathcal{A}$ before and after the removal of a set S of k nodes in G, respectively. Mathematically, X and Y are represented as $X = \mathcal{A}(G)$ and $Y = \mathcal{A}(G[V \backslash S])$, where $G[V \backslash S]$ is the subgraph induced by $V \backslash S$ on G. For any index $i = 1, ..., c_X$ and $j = 1, ..., c_Y$, let $x_i = |X_i|$, $y_j = |Y_j|$, and $n_{ij} = |X_i \cap Y_j|$. Finally, let $\bar{x} = \sum_{i=1}^{c_X} x_i$, $\bar{y} = \sum_{j=1}^{c_Y} y_j$ and $\bar{n} = \sum_{i=1}^{c_X} \sum_{j=1}^{c_Y} n_{ij}$ be the total size of communities in X and Y, and the total number of common nodes shared between X and Y, respectively.

(Normalized Mutual Information) In order to evaluate how much the network community structure changes before and after the removal of important nodes, we

utilize the concept of Normalized Mutual Information suggested in [20]. Basically, given two structures X and Y, $NMI(X, Y)$ is 1 if X and Y are identical and is 0 if X and Y are totally separated, and the higher the NMI score, the more similarity between X and Y. As a result, NMI is a well-suited metric dedicated for certifying the quality of community structures discovered by different detection algorithms. The effectiveness of this widely-accepted measure has also been extensively verified in the literature [55]. Formally, $NMI(X, Y)$ is defined as

$$NMI(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)},$$

where $H(X)$, $H(Y)$ and $I(X, Y)$ are the entropy of structures X and Y, and the Mutual Information conveyed between them, respectively. More details about NMI formulation will be elaborated in our analysis.

(Problem definition) Finally, the Community Structure Vulnerability (CSV) problem is formulated as follow.

Definition 1. *Given a network represented by an undirected and unweighted graph G, a specific community detection algorithm $\mathcal{A}$, and a positive integer $k \leq N$, we seek for a subset $S \subseteq V$ such that*

$$S = \underset{T \subseteq V, |T|=k}{\arg\min} \{NMI(\mathcal{A}(G), \mathcal{A}(G[V \setminus T]))\}.$$

In other words, CSV problem seeks for a subset $S \subseteq V$ of k nodes whose removal results in the maximum difference between the initial community structure X and the new community structure Y detected by $\mathcal{A}$ on $G[V \setminus S]$. We call S the Node-Vulnerability set of G since its removal maximally transforms network communities of G to different structures.

Remark. The formulation of CSV requires the community detection algorithm $\mathcal{A}$ as an input parameter. Because there is not yet an universal agreement or accepted definition of a network community, this input is necessary in the sense that different

algorithms with different objective functions might favor different sets of nodes, and thus, a good solution set for one community detection algorithm may not be good for the others. However, when there is a clear objective function for finding community structure, such as maximizing Modularity Q [55] or the total internal density [80], this requirement can be lifted. Nevertheless, the node selection strategy that relies more on the input network and less on the community detection algorithm is always of desire.

8.3 Analysis of NMI Measure

In this section, we investigate the possible conditions on sizes and the number of communities that can potentially lead to either the global or local minimization of $NMI(X, Y)$. We stress that these conditions are by no means universal or exhaustive since some of them might not hold true simultaneously, given the input parameters. Indeed, what we hope for is these conditions would provide us key insights into the selection of important nodes to maximally separate X and Y. In the coming paragraphs, we first discuss the NMI formulation in a greater detail, and then analyze it in terms of both disjoint and overlapping community structures.

8.3.1 NMI formulation

To evaluate $NMI(X, Y)$ [20] where $X = \{X_1, X_2, \ldots, X_{c_X}\}$ and $Y = \{Y_1, Y_2, \ldots, Y_{c_Y}\}$, we start out by considering community assignments X_i and Y_j, where X_i and Y_j indicate the community labels of a node t in X and Y, respectively. Without loss of generality, we can aslo assume that the labels X_i and Y_j are also values of two random "variables" X and Y (here we reuse notations X and Y to denote the two random variables), with joint distribution

$$P(X_i, Y_j) = P(X = X_i; Y = Y_i) = n_{ij}/(N - k),$$

and individual distribution

$$P(X_i) = P(X = X_i) = x_i/N,$$

$$P(Y_j) = P(Y = Y_j) = y_j/(N - k).$$

The entropy (or uncertainty) of X and Y is defined as [18]

$$H(X) = -\sum_{i=1}^{c_X} P(X_i)\log P(X_i) = -\sum_{i=1}^{c_X} \frac{x_i}{N}\log\frac{x_i}{N},$$

$$\begin{aligned} H(Y) &= -\sum_{j=1}^{c_Y} P(Y_j)\log P(Y_j) = -\sum_{j=1}^{c_Y} \frac{y_j}{N-k}\log\frac{y_j}{N-k} \\ &= \frac{1}{N-k}\big(\bar{y}\log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j\big). \end{aligned}$$

Note that in CSV problem, X can be derived straightforwardly based on $\mathcal{A}$ and G, and thus, quantities x_i's can also be inferred from these input parameters. Therefore, we simply consider x_i's and $H(X)$ as constants in this paper.

The Mutual Information $I(X, Y)$ [18] of two random variables X and Y is defined as

$$\begin{aligned} I(X, Y) &= \sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} P(X_i, Y_j)\log\frac{P(X_i, Y_j)}{P(X_i)P(Y_j)} \\ &= \sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} \frac{n_{ij}}{(N-k)}\log\frac{Nn_{ij}}{x_i y_j}. \end{aligned}$$

This measure is symmetric and it tells us how much we know about variable (or structure) Y if we already know about variable X, and vice versa. However, as indicated in [20][55], Mutual Information itself is not ideal as a global similarity metric since any subpartition of a given community structure X would result in the same mutual information with X, even though they can possibly be very different from each other. As a result, [20] introduces the Normalized Mutual Information which can overcome that limitation. Formally, NMI of two random variables X and Y is defined as

$$NMI(X, Y) = \frac{2I(X, Y)}{H(X) + H(Y)} \tag{8–1}$$

In term of notations, $NMI(X, Y)$ can be written as

$$\frac{2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} n_{ij}\log\frac{Nn_{ij}}{x_i y_j}}{(N-k)H(X) + \bar{y}\log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j} \tag{8–2}$$

8.3.2 Minimizing NMI in a disjoint community structure

When network communities are disjoint from each other, we have $X_i \cap X_s = \emptyset$, $\cup_{i=1}^{c_X} X_i = V$, $Y_j \cap Y_t = \emptyset$, and $\cup_{j=1}^{c_Y} Y_j = V \backslash S$ for all $i, s = 1, \ldots, c_X$ and all $j, t = 1, \ldots, c_Y$. As a result, the following equalities hold true: $\bar{x} = \sum_{i=1}^{c_X} x_i = N$, $\bar{y} = \sum_{j=1}^{c_Y} y_j = N - k$ and $\bar{n} = \sum_{ij} n_{ij} = N - k \quad (*)$.

8.3.2.1 Minimizing NMI within a community

We first investigate the behavior of $NMI(X, Y)$ in a special case where only one specific community of X is affected by the removal of set S of k nodes while other communities stay intact. We can assume that X_1 is the targeted community which is further divided into p smaller subcommunities of sizes $s_1, s_2, \ldots, s_p$ satisfying $\sum_{j=1}^{p} s_j = x_1 - k$. In this case

$$\begin{aligned} H(Y) &= \sum_{j=1}^{p} \frac{s_j}{N-k} \log \frac{N-k}{s_j} + \sum_{i=2}^{c_x} \frac{x_i}{N-k} \log \frac{N-k}{x_i} \\ &= \frac{(x_1 - k)\log(N-k) + \sum_{i=2}^{c_X} x_i \log \frac{N-k}{x_i} - \sum_{j=1}^{p} s_j \log s_j}{N-k}, \end{aligned}$$

and

$$\begin{aligned} I(X, Y) &= \sum_{j=1}^{p} \frac{s_j}{N-k} \log \frac{N}{x_1} + \sum_{i=2}^{c_x} \frac{x_i}{N-k} \log \frac{N}{x_i} \\ &= \frac{x_1 - k}{N-k} \log \frac{N}{x_1} + \sum_{i=2}^{c_X} \frac{x_i}{N-k} \log \frac{N}{x_i}. \end{aligned}$$

Thus, $NMI(X, Y)$ is minimized when $\sum_{j=1}^{p} s_j \log s_j$ is minimized. Since function $s \log s$ is strictly convex for any $s > 0$, we apply Jensen's inequality [18] to this summation and get

$$\frac{1}{p} \sum_{j=1}^{p} s_j \log s_j \geq \frac{\sum_{j=1}^{p} s_j}{p} \log \frac{\sum_{j=1}^{p} s_j}{p} = \frac{x_1}{p} \log \frac{x_1}{p},$$

with the equality holds when all s_j's are equal to each other. It reveals from this inequality that, in order to further minimize the RHS quantity, one can try to break X_1 into as many smaller communities of the relatively same size as possible (i.e., to enlarge

p as much as possible while ensuring s_i's are all equal). This intuition makes senses since a new structure of X_1 with all singleton communities will incur $\sum_{j=1}^{p} s_j \log s_j = 0$, and hence, will maximize $H(Y)$ and in turn will minimize $NMI(X, Y)$. However, since the new structure of X_1 depends on the community detection algorithm $\mathcal{A}$, the all-singleton communities scenario might not always be the case. Furthermore, will this crucial observation hold true in a general disjoint and overlapping community structure? We tend to lean over the affirmative answer through our analysis in the coming subsections.

8.3.2.2 Minimizing NMI in a general disjoint community structure

In general disjoint community structure, the equalities $(*)$ help to simplify $NMI(X, Y)$ (eq. 8–2) to

$$\frac{2\sum_{i=1}^{c_X}\sum_{j=1}^{c_Y} n_{ij} \log \frac{Nn_{ij}}{x_i y_j}}{(N-k)H(X) + (N-k)\log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j}.$$

In order to minimize the above ratio, one would seek for the conditions in which the numerator of $NMI(X, Y)$ is minimized while its denominator is also maximized. To maximize the latter quantity, we need to minimize $\sum_{j=1}^{c_Y} y_j \log y_j$. Applying Jensen's inequality to this summand gives

$$\frac{1}{c_Y}\sum_{j=1}^{c_Y} y_j \log y_j \geq \frac{\bar{y}}{c_Y} \log \frac{\bar{y}}{c_Y} = \frac{N-k}{c_Y} \log \frac{N-k}{c_Y},$$

and thus $\sum_{j=1}^{c_Y} y_j \log y_j$ can attain it minimum at $(N-k)\log \frac{N-k}{c_Y}$ with equality holds when all y_j's are equal to each other. As N and k are input parameters, $\log \frac{N-k}{c_Y}$ can further be minimized when c_Y is as large as possible, while requiring y_j's to be equal to each other. Mathematically, this can be achieved when Y contains exactly $c_Y \equiv (N-k)$ singleton communities. However, since our problem depends on the detection algorithm, this inequality advises that the newly community structure Y should contain as many communities of relatively the same size as possible. We take into account this observation as it will play a key role in our important-node selection process. This observation is also coincident with what inferred in the prior special case, and intuitively agrees with the concept of Critical Node Detection (CND) [25] and Balanced Graph

Partitioning (BGP) [2] whose goals aim to delete nodes and cut the input graph into p connected components of relatively the same size. However, CSV fundamentally differs from these problems in the senses that connected components in BGP and CND do not necessarily reflex network communities.

In order to minimize the numerator, we rewrite it as

$$I(X, Y) = \frac{1}{N-k}\left(\sum_{ij} n_{ij} \log \frac{N n_{ij}}{y_j} - \sum_{ij} n_{ij} \log x_i\right).$$

Applying Log Sum Theorem [18] to the first summand gives

$$\begin{aligned} I(X, Y) &\geq \frac{1}{N-k}\left(\bar{n} \log \frac{N\bar{n}}{c_X \bar{y}} - \sum_{ij} n_{ij} \log x_i\right) \\ &= \log \frac{N}{c_X} - \frac{1}{N-k}\sum_{i} (x_i - l_i) \log x_i, \end{aligned}$$

because $\bar{n} = \bar{y} = N - k$ and $\sum_{j=1}^{c_Y} n_{ij} = x_i - l_i, \forall i = 1, \ldots, c_X$, where l_i is the number of deleted (or lost) nodes in community X_i, and l_i's satisfy $\sum_{i=1}^{c_X} l_i = k$. The equality holds when n_{ij}/y_j is a constant, say $\gamma \geq 0$, for all $i = 1, \ldots, c_X, j = 1, \ldots, c_Y$. If we assume that this is the case, then $\sum_{j=1}^{c_Y} n_{ij} = \gamma \sum_{j=1}^{c_Y} y_j = \gamma(N-k)$, which in turn implies $N - k = \sum_{ij} n_{ij} = c_X \gamma (N-k)$. Hence, $\gamma = 1/c_X$ and thus, $l_i = x_i - (N-k)/c_X$. Therefore, to minimize the second summand, the equation $l_i = x_i - (N-k)/c_X$ advises that we should put more focus on (i.e., remove more nodes in) big-sized communities X_i of X to break it into smaller modules. This breaking down of big-sized communities partially supports the prior observation that communities of Y should have relatively the same size. Note that in this analysis, we have assumed that n_{ij}/y_j is a constant for all pair of i and j. In practice, this might not always be the case since real communities can be distributed differently based on the underlying detection algorithm. Nevertheless, we find this observation helpful as it suggests a general instruction for selecting important nodes in the network.

8.3.3 Minimizing NMI in an overlapped community structure

The minimization of $NMI(X, Y)$ measure is much more complicated when network communities can overlap with each other. In particular, the conditions $\cup_{i=1}^{c_X} X_i = V$ and $\cup_{j=1}^{c_Y} Y_j = V \backslash S$ still hold in this case; however, $X_i \cap X_s$ and $Y_j \cap Y_t$ might not be empty for some $i, s = 1, \ldots, c_X$ and $j, t = 1, \ldots, c_Y$. These facts indicate that $\bar{x} = \sum_{i=1}^{c_X} x_i \geq N$, $\bar{y} = \sum_{j=1}^{c_Y} y_j \geq N - k$ and $\bar{n} = \sum_{ij} n_{ij} \geq N - k$.

Our analysis strategy in this case is similar to the prior one as we also strive for maximizing the denominator while minimizing the numerator of $NMI(X, Y)$ (eq. 8–2). Because $\bar{n} \geq N - k$, the minimization of the top term $I(X, Y)$ no longer depends only on x_i's anymore. One way to work around this issue is to investigate the relative correlation between the total community size $\bar{y}$ and the number of communities c_Y. Let $\alpha_{\mathcal{A}} = \frac{\bar{y}}{c_Y}$ be the ratio between these two quantities, or in other words, the averaged community size. The denominator of $NMI(X, Y)$ is evaluated as

$$\begin{aligned} \bar{y} \log(N-k) - \sum_{j=1}^{c_Y} y_j \log y_j &\leq \bar{y} \left(\log(N-k) - \frac{\log(\bar{y}/c_Y)}{c_Y} \right) \\ &= \bar{y} \log (N-k) - \alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}. \end{aligned}$$

with equality holds when all y_j's are equal to each other. To further maximize this denominator, we need $\bar{y}$ to be as large as possible while keeping $\alpha_{\mathcal{A}}$ as small as possible, i.e., the new community structure Y should contain more and more communities as to increase c_Y as well as to lower down $\alpha_{\mathcal{A}}$.

Due to the dependence on the specific detection algorithm $\mathcal{A}$, this optimization on the correlation between $\bar{y}$ and c_Y might not be globally achieved. However, a coarse analysis between $\bar{y}$ and c_Y can relatively be conducted in the following senses: if we assume that $\bar{y}$ is within a constant factor of the total number of actual nodes $(N - k)$, i.e., $\bar{y} \leq a_0(N - k)$ for some constant $a_0 > 1$, we can then increase the value of the RHS by breaking as many communities as possible while keeping them having the size (i.e., enlarge c_Y and keep y_j's are all the same), which helps to reduce the impact of

$\alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}$. This observation, though relative, agrees with what we achieved in the case of disjoint community structure. In an unfortunate case where $\bar{y}$ is not known to be within any constant factor of $(N - k)$, the observation might not hold since both $\bar{y}$ and c_Y can be arbitrary large and thus, $\alpha_{\mathcal{A}} \log \alpha_{\mathcal{A}}$ could still be relatively small.

Next, applying Log Sum Theorem on the numerator yields

$$I(X, Y) = \sum_{ij} n_{ij} \log \frac{N n_{ij}}{x_i y_j} \geq \bar{n} \log \frac{N \bar{n}}{\bar{x} \bar{y}},$$

with equality holds when $\frac{N n_{ij}}{x_i y_j}$ is a constant for all $i = 1, \ldots, c_X$ and $j = 1, \ldots, c_Y$. Thus, one can try to minimize $I(X, Y)$ by deleting nodes in such a way that $\bar{n}$ is maximized and $\bar{y}$ is minimized while making sure that $\frac{N n_{ij}}{x_i y_j}$ is a constant. As a result, this minimization of $I(X, Y)$ is a multiple-objective optimizations problem which may not have a feasible solution. However, if we assume that the later condition is imposed, i.e., $\frac{N n_{ij}}{x_i y_j} = \beta_{\mathcal{A}}$ for some constant $\beta_{\mathcal{A}} > 0$, then $n_{ij} = \frac{\beta_{\mathcal{A}} x_i y_j}{N}$, and thus $\bar{n} = \frac{\beta_{\mathcal{A}}}{N} \bar{x} \bar{y}$. This reduces the above inequality to

$$I(X, Y) \geq \frac{\bar{x}}{N} \beta_{\mathcal{A}} \bar{y} \log \beta_{\mathcal{A}} N.$$

The RHS of the inequality advises that, in order to minimized $I(X, Y)$, the total size of network communities should not be too large while the overlapping ratio of every community should be equal to each other and be as small as possible. This is a different criterion from the disjoint community structure point of view.

8.4 A Solution to CSV Problem

In the following paragraphs, we consider the scenario when maximizing the internal density [80] is the objective function for finding network communities, i.e., communities of G are assumed to have optimized internal densities. In this manner, we present genEdeg, an algorithm for solving CSV problem that is independent of the underlying community detection algorithm $\mathcal{A}$. Our solution strategy will try to break larger communities to as many small ones as possible while looking for them to have the relatively same size with small overlapping ratios. The idea of our strategy is based

on the following intuition: since communities in X are optimized for their internal density, they are likely to contain strong substructures that are tightly connected which form the cores of these communities. As a result, the removal of crucial nodes in a core might potentially break the community into smaller modules. Moreover, as nodes in a core are tightly connected, there should be some edge that generate them, i.e., all nodes in the core are incident to both endpoints of this edge. Inspired by this intuition, our strategy works towards the identification of these generating edges of a community, and then seek for the minimum set of generating edges that composes the original communities.

Let D be a subset of V. Denote by $\Psi(D) = \frac{2m_D}{n_D(n_D-1)}$ the internal density of D and by $\tau(D) = \frac{n_D(n_D-1) - \frac{2}{n_D(n_D-1)}}{2}$ the threshold function on the internal density of D, respectively. For any nodes $u, v \in D$, if edge (u, v) is not in E, we call it a missing edge in D. In addition, we call an edge in D "negative" if it is incident to a missing edge in D, and "positive" otherwise. We define the concept of generating edges of D as follow

Definition 2. *(Generating edge) For any edge (u, v) in D, if $D = (D \cap N(u) \cap N(v)) \cup \{u, v\}$ and $\Psi(D) \geq \tau(D)$, we call (u, v) a generating edge of D. We further call D a local core generated by (u, v) and write $gen(u, v) = D$.*

For any community C of G, a set $L \subseteq E$ is called a "generating edge set" of a C if $\cup_{(u,v)\in L} gen(u, v) = C$. Since C can be generated by different generating edge sets and we are constrained on the node budget, we would intuitively seek for the generating edge set of minimal cardinality.

Definition 3. *(Minimum Generating Edge Set) Given a community C of G, the MGES problem seeks for a generating edge set L^* of C with the smallest cardinality.*

The cores generated by edges in a MGES of a community C of G are tightly connected and they all together compose C. As a result, if we delete an endpoint of every edge in a MGES, C will be broken into smaller modules with the number of modules is at least the number of edges in a MGES (Lemma 16). Since our goal is to break the current community structure X into as many new communities as possible,

the removal of crucial nodes defined by edges in a MGES will be a good heuristic for this purpose. But first and foremost, we need to characterize all MGESs in the current community structure X based only on the input network G. Lemma 17 realizes the location of the generating edge(s) of a local core in a community C: they have to adjacent to nodes with the highest degree in C. Based on this result, we present in Alg. 18 a procedure that can correctly find the MGES of a given community C (Theorem 8.1).

Algorithm 18 An optimal algorithm for finding the MGES

Input: Network $G = (V, E)$ and a community $C \in X$;
Output: Minimum generating edge set L^* of C;

0. Mark all nodes as "unassigned" and $L^* = \emptyset$.
1. Remove all negative edges in C. If any edge(s) survive, they are candidate for generating edges in their corresponding communities, including them to L^*, go to step 2. Else, go to step 3.
2. Reconstruct local cores based on generating edges found in step 1. Mark all nodes in those communities as "assigned". Discard generating edges in L^* that fall into any newly constructed communities. Return if all edges are assigned.
3. Find the set U as in Lemma 17. Find the edge in $NE(U)$ that can generate a local community having the largest size. Include this edge to L^* and mark all nodes in the new local community as "assigned". Ties are broken randomly. Return if all edges are assigned.
4. If there are still unassigned nodes, say the set $I \subseteq C$, construct $G_1 = G[(I \cup N(I)) \cap C]$. Go to back to step 1.

Lemma 16. *Let L^* be a MSGE of a community C. The removal of an endpoint in every edge of L^* will break C into at least $|L^*|$ subcommunities.*

Proof. Clearly, the removal of an endpoint of every edge in L^* will degrade the internal density of each core since the endpoint of the generating edge is of full degree in its core. Now, if the number of subcommunities resulted in the node removal is less than $|L^*|$, it means there are at least two cores that are merged together. That is there are cores c_1 and c_2 are merged together even with less internal density. This should not be the case since otherwise, they have to be identified as a single core at the first place.

Their combination, as a result, implies that C has a MGES of size less than $|L^*|$, which raises a contradiction to the assumption that L^* is a MGES of C. □

Lemma 17. *Let C be a subset of V, $U = \{u \in C | d_u^C$ is the highest in $C\}$ and $NE(U) = \{(u, v) | u \in U$ or $v \in U$ but not both $\}$. Then, $|NE(U) \cap L^*| \geq 1$.*

Proof. After each refreshment in step 2, let u be the node with the highest indegree in C. After step 1 of Alg. 18, all negative edges are deleted since they do not contribute to the actual generating set L^*. As such, edges incident to u are not negative. This in turn implies that they are candidates for generating edges. Now, iterate through all edges incident to u and choose the one that generates the biggest-sized core. This edge should be in the list L^*. □

Theorem 8.1. *Let d_C be the maximum in-degree of a node in C. Alg. 18 takes $O(d_C|C|)$ time in the worst case scenarios and returns an optimal solution for MGES problem.*

Proof. Since every time Lemma 17 makes sure that at least one edge should be added to L^* and the procedure terminates when no edges left, the Alg. 18 should terminate. Moreover, it is verifiable that Alg. 18 take at time as most the number of edges in C, which is $O(d_C|C|)$. Also, due to the intense internal density of a core, every time an edge is added into L^*, that edge actually generates the largest core possible. The proof follows from this fact, Lemma 17 and the exhaustive property of Alg. 18. □

Algorithm 19 *genEdge* - A node selection strategy for CSV based on generating edges

---

Input: Network $G = (V, E)$, $X = \mathcal{A}(G)$;
Output: A set $S \subseteq V$ of k nodes;

1. Use Alg. 18 to find $L^*_{X_i}$ for all communities X_i's in X.
2. Sort all communities X_i's in X by their sizes of MGSEs.
3. Sort all nodes in G by the number of generating edges that they are incident to in X_i. If there is a tie, sort them by their degrees in G.
4. Return top k nodes in step 3.

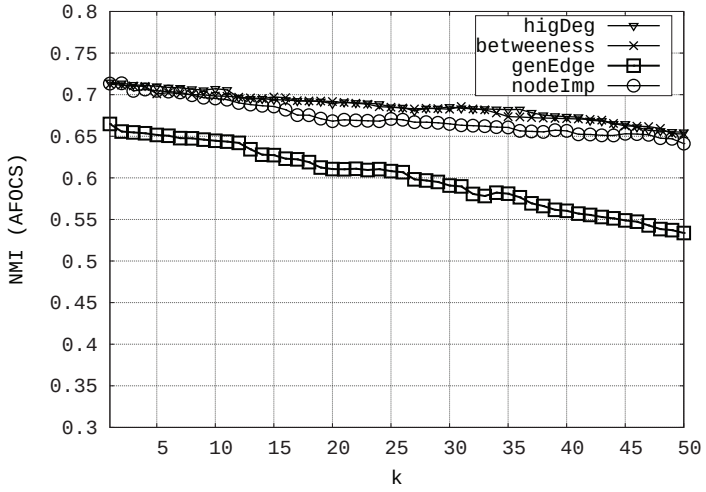
With the optimal solution of MGES taken into account, we next suggest a heuristic for selecting important nodes following the guidelines suggested in the previous. In

particular, our heuristic selects nodes in a greedy manner, starting from communities that have large-size MGESs. Moreover, in the MGES of each community C, we give priority to nodes that are incident to more generating edges since their removals will break C into more subcommunities.

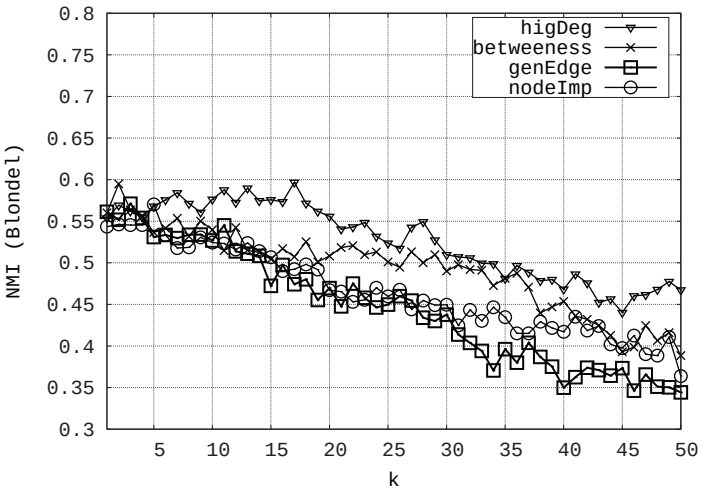
8.5 Experimental Results

In this section, we show the empirical results of our node selection strategy for CSV on both synthesized networks with known community structures and real-world social traces including the Reality mining cellular dataset [29], Facebook [100] and Foursquare [21] social networks. In order to certify the performance of our approach, we compare the results obtained by the following methods: High degree centrality ($highDeg$) selects top k nodes in G with the highest degrees, $betweeness$ centrality ($betweeness$) selects top k nodes in G with the highest $betweeness$es (where the $betweeness$ of a node u is the number of shortest paths in G that pass through u), Generating edges ($genEdge$) - our strategy described in Alg. 19, and finally, Node Importance ($nodeImp$) [105] selects top k nodes by their importance to the community structure.

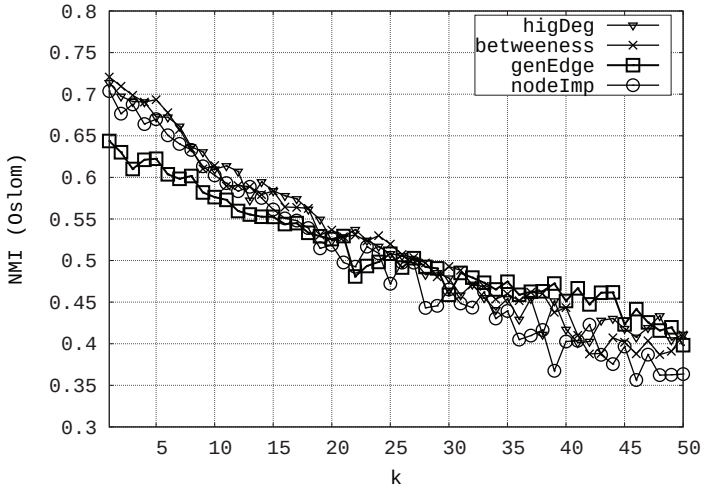
We first examine the effect of the underlying community detection methods by comparing results obtained by AFOCS [80], Blondel [6] and Oslom [61] algorithms to the embedded groundtruths. In particular, we set X to be the groundtruth community structure and when S is removed from the network $NMI(X, Y)$ is reported, where $Y = AFOCS(G[V \backslash S])$, $Y = Blondel(G[V \backslash S])$ and $Y = Oslom(G[V \backslash S])$, respectively. These methods have been empirically certified in the literature to the best algorithms for finding non-overlapping and overlapping community structure [55]. Verifying our strategy on synthesized networks not only certifies its performance but also provides us the confidence to its behaviors when applied to real-word traces. We next demonstrate the following quantities (1) the NMI differences between community structures before and after the node removal, which is our main objective function, (2) the number of



A NMI scores by AFOCS

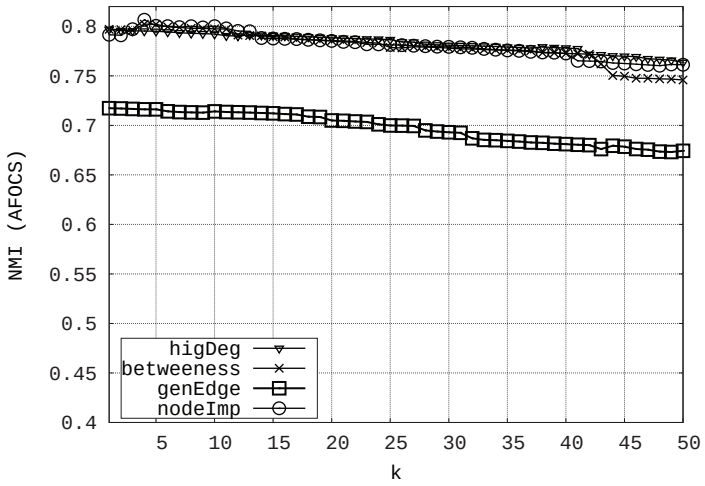


B NMI scores by Blondel

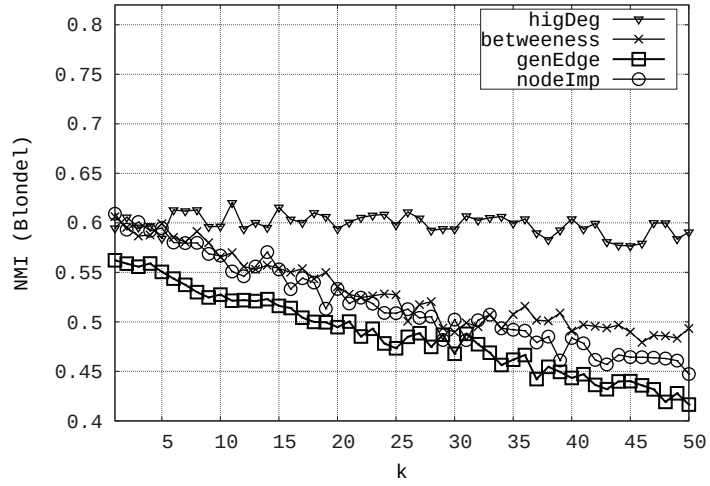


C NMI scores by Oslom

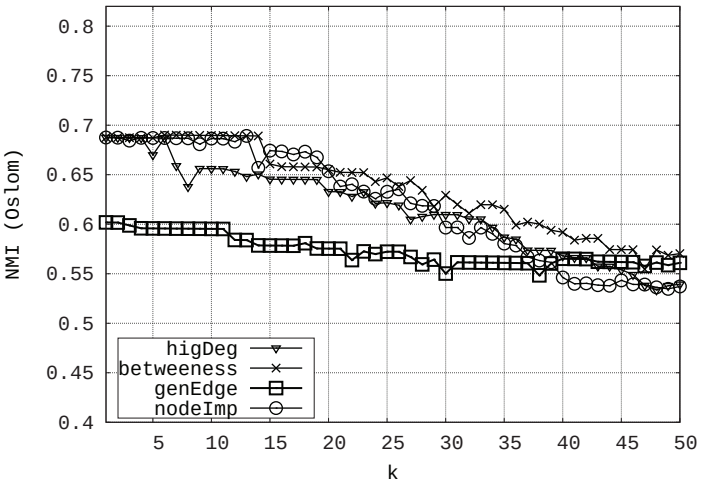
Figure 8-1. Comparison among different node selection strategies on synthesized networks with $N = 2500$ nodes



A NMI scores on AFOCS



B NMI scores on Blondel



C NMI scores on Oslom

Figure 8-2. Comparison among different node selection strategies on synthesized networks with $N = 5000$ nodes

communities in the new structure, and (3) the average size of the network communities in the new structure.

8.5.1 Results on synthesized networks

Set up: We use the well-known LFR overlapping benchmark [55] to generate test networks. The number of nodes are $N = 2500$ and 5000, the mixing parameter $\mu = 0.15$, the community sizes $c_{min} = 10$ and $c_{max} = 50$ for $N = 2500$ and $c_{min} = 30$ and $c_{max} = 100$ for $N = 5000$. At every k nodes are removed from the network, the network community structure is reidentified and compared to the original embedded one (or the ground-truth). The overlapping threshold β in AFOCS is set at 0.7 and all tests are averaged on 100 runs for consistency.

8.5.1.1 Solution quality

We first evaluate the performance of all aforementioned node selections strategies on different community detection algorithms $AFCOS$, Blondel and Oslom, respectively. Because the ground-truth communities on synthesized networks are given a priori, comparisons through NMI scores among these strategies as well as among detection algorithms are therefore valid, and the lower NMI scores a strategy obtains, the more effective it seems to be. In addition, the higher the remaining NMI measure a detection algorithm obtains after the node removal, the more resistant to node vulnerability it seems to be.

The quality of node selection solutions, are reported in figures 8-1 and 8-2. In a general trend, NMI scores tend to drop down quickly as more nodes are removed from the network when $N = 2500$; however, they degrade much slower in networks with $N = 5000$. The first observation revealed in those figures is that our approach $genEdge$ achieves the best (lowest) NMI scores on almost all test cases. In average, on networks with 2500 nodes, $genEdge$ is 14% better than both $highDeg$ and $betweeness$, and is 12% better than $nodeImp$ on AFOCS algorithm; and is 19%, 11% and 5% better than $highDeg$, $betweeness$, and $nodeImp$ on Blondel algorithm (figure 8-1A, 8-1B). On Oslom algorithm,

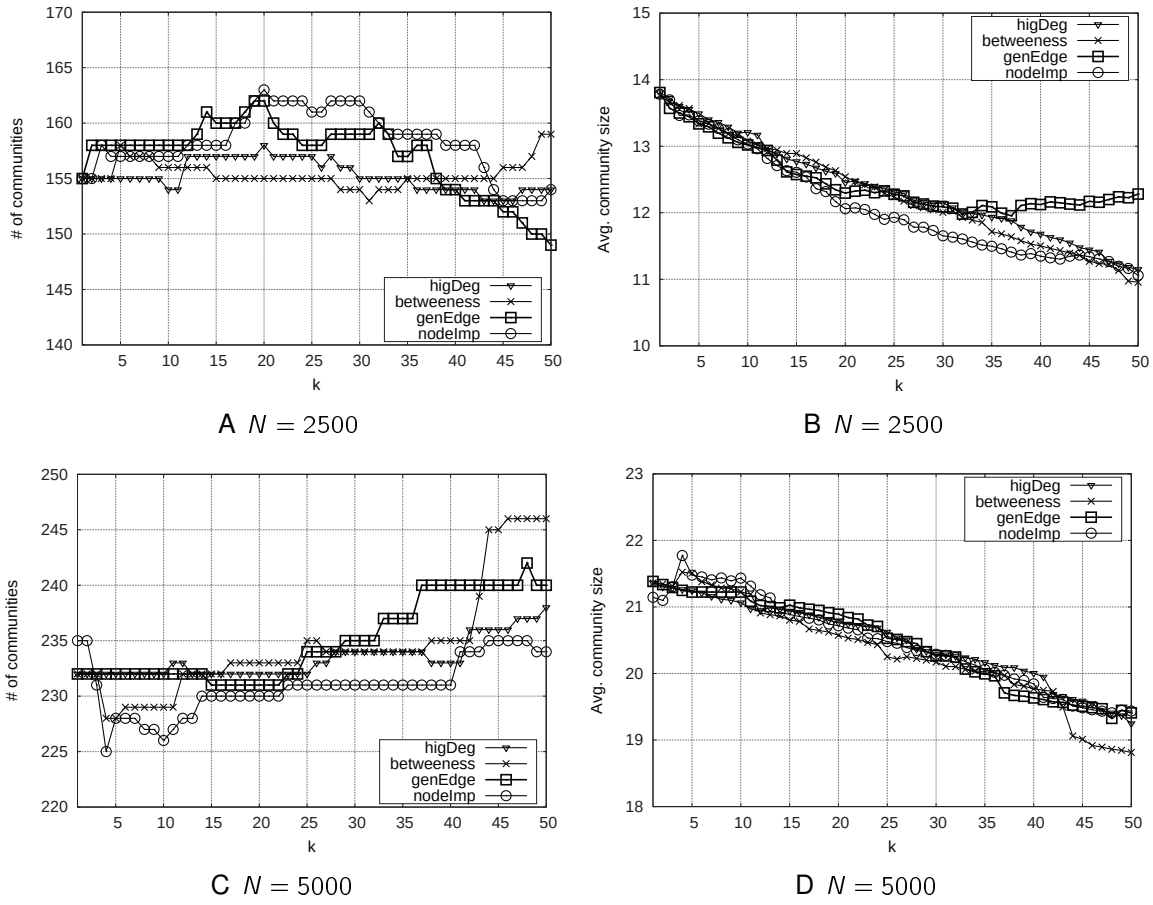


Figure 8-3. Results obtained by AFOCS on networks with $N = 2500$ nodes and $N = 2500$ nodes.

genEdge differs insignificant with *highDeg* and *betweeness* with 1.5% and 1.4% better, and is only lagged behind *nodeImp* with 3% lower NMI scores. On network with 5000 nodes, *genEdge* still outperforms other strategies with 12% lower NMI scores than the others on AFOCS algorithm, and with 23%, 8% and 6% lower NMI scores than *highDeg*, *betweeness* and *nodeImp* on Blondel algorithm, and finally, with 7%, 10% and 8% better than the others on Oslom algorithm (figure 8-2). These results imply that *genEdge* node selection strategy performs excellently with competitive results on different community detection algorithm in comparison with other strategies.

The second observation we obtain from figures 8-1 and 8-2 is that the top-of-the-list node seems to be essential to the network community structure. The removal of only

this node from the network brings the NMI scores to as low as 0.7 - 0.8 on AFOCS (figure 8-1A, 8-2A), to 0.58 - 0.6 on Blondel algorithm (figure 8-1B, 8-2B), and to 0.7 on Oslom algorithm. Furthermore, the top 15-20 nodes are also vital to the network community structure detected by Oslom and Blondel since their destruction brings the NMI scores down to 0.5, the threshold where the community structure become stochastic and fuzzy to recognize. The NMI values on AFOCS algorithm, on the other hand, do not suffer from this destruction as they only come close to 0.5 when almost $k = 50$ nodes are removed from the networks with $N = 2500$ nodes (figure 8-1A).

Finally, the last observation inferred from figures 8-1 and 8-2 is that, among the three community detection algorithms, AFOCS algorithm obtains the highest remaining NMI values when the same number of nodes is removed from the networks. In other words, AFOCS was able to detect the community structure which was of the most similarity to the ground-truth communities. As we discussed above, this observation implies that AFOCS seems to be the detection algorithm which is more resistant to node vulnerability than the other algorithms. Therefore, we employ AFOCS as the main community detection algorithm to further analyze network communities of real-world traces.

8.5.1.2 The number of communities and their sizes

We next examine the number of communities and their sizes when k important nodes are removed from the network. As discussed in subsection 8.4, our selection strategy gives priority to breaking the current community structure into more communities while looking for their sizes to be relatively the same in order to minimize NMI measure. The results are presented in figure 8-3.

As reported in these figures, the numbers of new communities generated by $genEdge$ tend to increase as more nodes are excluded; however, they differ insignificantly from other methods on small networks of 2500 nodes (figure 8-3A), but the differences become more visible on larger networks of 5000 nodes (figure 8-3C). In particular, the

Table 8-1. Statistic of social traces

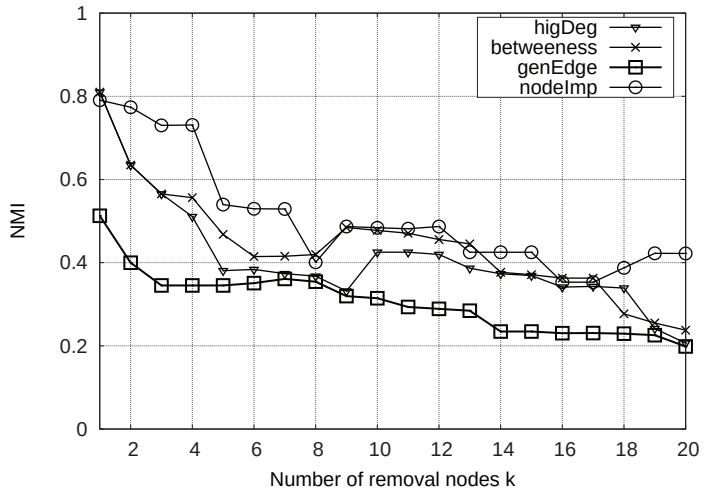
Data	N	M	Avg. Deg	Max. Com. Size
Reality	100	3100	62	35
Facebook	63731	1.5M	23.50	33425
Foursquare	47260	1.1M	49.13	30381

number of communities generated by *genEdge* is the second highest when $N = 5000$ (only below *betweeness* method) while the average sizes of communities are relatively equal to other methods (figure 8-3B and 8-3D). One might question why the NMI scores returned by *genEdge* is still high since its number of communities and average community size are relatively the same as the other. One possible reason is because new communities formed by other strategies might possibly be the subcommunities or parts of of the original structure, which in turn results in high similarity to the ground-truth. Our strategy, on the other hand, makes sure that once a node incident to the most generating edges is excluded, the subcommunity structure is broken and the new community structure has little similarity to the original one, and hence, the lower NMI measures.

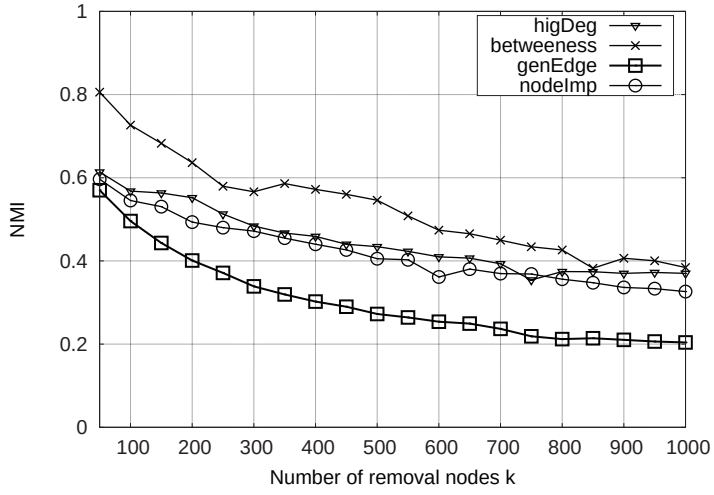
8.5.2 Results on real-world traces

We further present the empirical results of CSV on real-world networks including Reality mobile phone data [29], Facebook [100] and Foursquare [21] datasets. The overview of these datasets is summarized in Table 8-1.

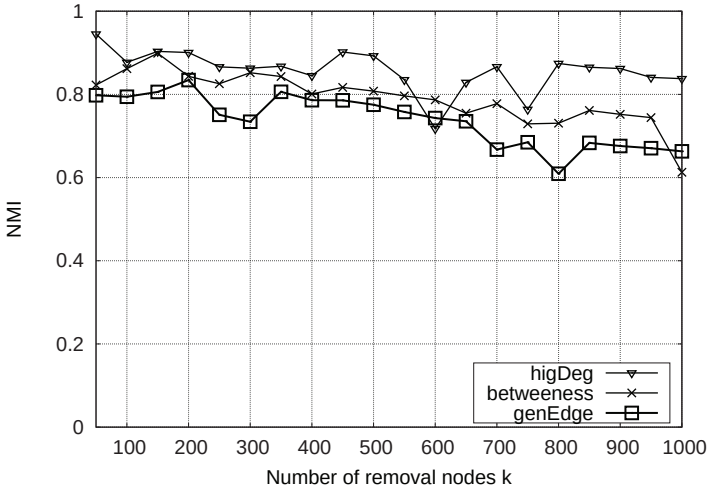
Reality Mining dataset provided by the MIT Media Lab. This dataset contains communication, proximity, location, call, and activity information from 100 students at MIT over the course of the 2004-2005 academic year. Facebook dataset contains friendship information (i.e., who is friend with whom and wall posts) among New Orleans regional network on Facebook, spanning from Sep 2006 to Jan 2009. To collect the information, the authors created several Facebook accounts, joined each



A Reality



B Foursquare



C Facebook

Figure 8-4. NMI scores on Reality mining data, Foursquare and Facebook networks obtained by AFOCS ($k = 50...1000$)

to the regional network, started crawling from a single user and visited all friends in a breath-first-search fashion. Foursquare dataset contains location and activities of 47260 users on Foursquare social network on May 2011 - Jul 2011. To collect the data, we created several Foursquare accounts, joined to the network, started crawling from a single user and visited all friends also in a breadth-first-search fashion.

On Reality Mining dataset, we set $k = 1...20$ and report result in figure 8-4A. It reveals from this figure that community structure in this dataset is extremely vulnerable to node attacks since the removal of only 2 nodes, found by $genEdge$ is enough to make the new community structure significantly differs from the original one as it brings down the NMI values to 0.4. In comparison with other node selection methods, $genEdge$ still perform excellently and is about 14% - 17% better than the others. We note that the first node identified by $genEdge$ is indeed crucial to the community structure of this network since it immediately brings down NMI score to 0.6 while the other does not seem to discover this important feature. Furthermore, when too many nodes are removed from the network, the network does seem to contain communities any more or the community structure become extremely fuzzy as NMI values converge down to around 0.2. This is understandable since this dataset is of small size with a very high average node degree.

On larger networks Facebook and Foursquare, we set k from 50 nodes to 1000 nodes (only 2.1% and 1.5% number of nodes of Foursquare and Facebook networks) with a 50-node increment at a time. The numerical results are reported in figure 8-4. In general, NMI values of all methods degrade quickly on Foursquare networks, and tend to decrease slower on Facebook networks. As more nodes are excluded from the network, $genEdge$ still achieves the best performance on both networks with significantly lower NMI values than the other methods. Specifically, on Foursquare with high average degree and internal community density, the removal of nodes incident to the most generating edges in $genEdge$ significantly leads to the separation of network community structure as NMI scores drop down to 0.2 in $genEdge$. On Facebook network, the

similarity between the original and new community structure seem to retain fairly high even all 1000 nodes are removed, whereas the new structure of ArXiv network is at the edge of stochastic threshold since the NMI measure is around 0.5. This implies that community structure in Foursquare network is also extremely vulnerable to node removal attacks, while the mature Facebook network does not seem to suffer this threat. One possible reason for this is since Facebook contains a giant community with low average degree, it therefore requires much more effort in order to break that giant community apart.

In summary, the experiments on both synthesized and real-work social network confirm the effectiveness of our proposed method based on generating edges. The empirical results also confirm that, $genEdge$ outperforms other heuristic methods on other community detection methods such as AFOCS, Blondel and Oslom algorithms.

8.6 An Application in DTNs

We present a practical application where the detection of overlapping network communities plays a vital role in forwarding strategies in communication networks. In order to evaluate the impact of community restructuring in complex networks, we compare the set of critical nodes identified by our community structure vulnerability algorithm to the set of nodes selected using aforementioned algorithms. Furthermore, in order to evaluate which one of the critical node set is the most critical, we study how the removal of the critical node set influences the performance of routing in Pocket Switched Networks (PSN), in terms of average message delivery ratio, and delivery-time.

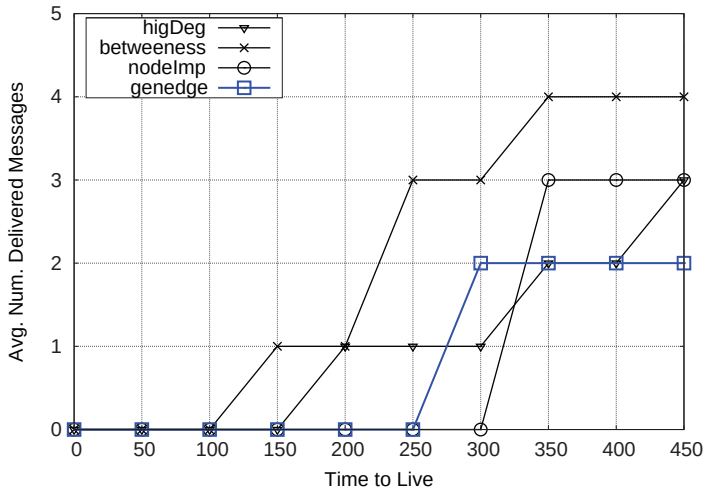
PSNs are a particular case of DTNs, where the nodes of the network correspond to actual people that are equipped with portable devices (i.e., mobile phones), and that use these portable devices to communicate. Because of the high degree of mobility of this type of networks, a path between a source and a destination seldom exists, therefore most of the approaches to routing in this kind of environments adopt a store-carry-and-forward approach. In store-carry-and-forward approaches, messages

are stored locally and, depending on the approach, they are forwarded or replicated to the encountered nodes when an opportunity occurs. In this manner, a node is important if it serves as a hub to forward the messages to other devices. As a result, the failures of these important nodes shall degrade the message delivery ratio while shall incur more duplicate messages and delivery time.

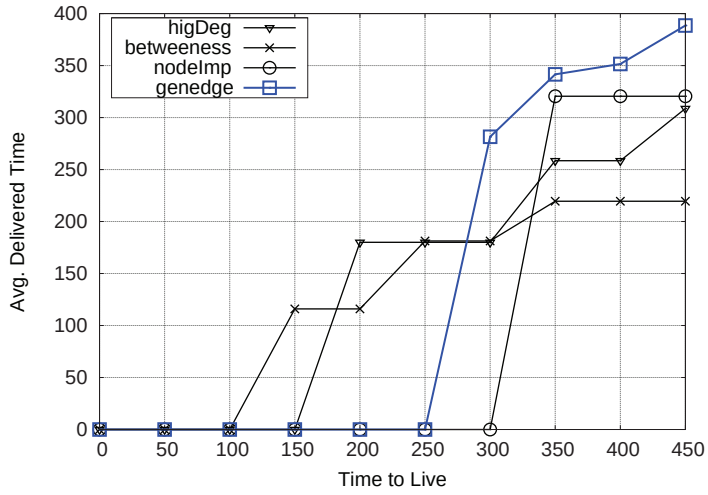
We use the HAGGLE dataset [94]. This trace was collected at the Infocom conference in 2006 in Barcelona. 70 students and researchers attending the workshop were equipped with iMote devices that registered they encounter for the duration of the conference (3 days). In addition to the 70 mobile partecipants, approximately 20 static, long range iMotes were deployed throughout the area of the conference. A total of 1000 messages are created and uniformly distributed during the experiment duration and each message can not exist longer than a threshold time-to-live. In our evaluation we will focus on the PSN routing algorithm inspired by BubbleRap [47]. While we expect the performance of this protocol to deteriorate upon the removal of important nodes, we expect the performances of BubbleRap to deteriorate more quickly, because of the reliance of the protocol on the community structure. Because BubbleRap relies on the knowledge of the community structure to route the messages, and because we realize that different algorithms that attempt to find the community structure use different objective functions that may be more susceptible to the removal of nodes, we consider evaluate the average delivery ratio, average delivery time and the average number of copied messages.

Results

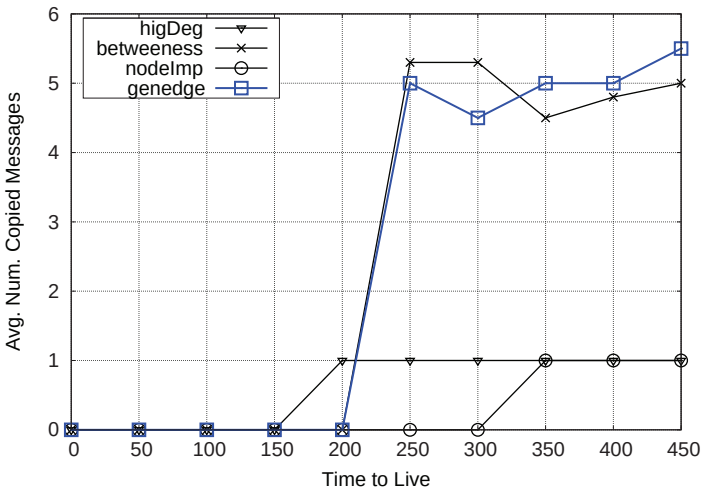
As the removal of 10 node in Haggle dataset is enough to make it original community structure to become stochastic (figure 8-6), we fix $k = 10$ and report the results as a function of $time - to - live$ (the amount of time a message can exist). The performances of all methods are presented in figure 8-5. As reported in subfigures 8-5A, 8-5B and 8-5C, the removal of nodes selected by genEdge approach significantly



A Avg. Delivered Messages



B Avg. Delivery Time



C Avg. Number of copied Messages

Figure 8-5. Simulation results on HAGGLE dataset.

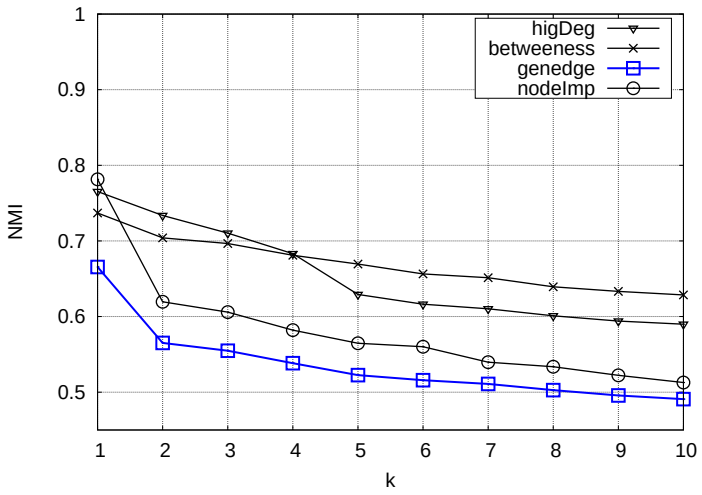


Figure 8-6. NMI measure on Haggle dataset.

degrades the performance of BubbleRap forwarding and routing system in terms of not only delivered messages and time but also the numbers of copied messages. As depicted in subfigure 8-5A, the averaged number of messages delivered by BubbleRap under genEdge and $time - to - live = 450s$ is only two, whereas those under highDeg, betweeness and $nodeImp$ are four, three and three, which implies 100% and 50% system downgrade when only 10 nodes are excluded from the networks. This also means nodes selected by genEdge are of important role in maintaining the normal operation of the whole network. Furthermore, when nodes are removed from the network, one expects that the delivery time should be increased as a consequnce because participants now have less chances to communicate with each other, and thus, it should take longer for participating devices to forward the carried messages. This intuition is nicely reflected in figure 8-5B. As reported in this subfigure, the average amount of time required to deliver carried messages increases significantly as $time - to - live$ increases (note that from 0-100s, there was no message delivered, and thus, the delivery time was 0). In terms of delivery time, the removal of nodes under genEdge affects the system to requires a huge extra amount to deliver the messages in comparison with other methods. In particular, the system delivery time under genEdge is

about 1.25x, 1.7x and 1.21x higher than that under betweeness, $nodeImp$ and highDeg when $time - to - live = 450$. Moreover, the number of copied messages, affected by genEdge approach, is also the highest one among other methods. This means that genEdge heuristic algorithm, indeed, selects appropriate nodes whose effects significantly reduce the system performance as reported by the three evaluated factors.

CHAPTER 9
CONCLUSIONS

In this dissertation, we establish the fundamental knowledge on the following aspects of the complex network science (1) the network organizational principals via the discovery of its dynamic community structure (2) the assessment of the community structure vulnerability, and (3) the social-based solutions for practical applications enabled by complex systems, such as in online social networks and mobile networks. We suggested two adaptive frameworks for discovering the dynamic network community structure and analyze theoretical results that guarantee their performances. In the execution perspective, our methods are adaptive, and thus, are scalable for very large networks with very competitive experimental results.

To investigate the assessment of the network community structure vulnerability, we introduce the new problem of identifying key nodes whose removal can maximally reform the current network communities. Those nodes are important in maintaining the normal functioning of the whole system, such as in the case of DTNs (in a mobile network) or lung cancer (in a biological network). Our work presents first and preliminary yet important insights, in terms of both theoretical results and heuristic algorithms, into the vulnerability assessment of the network community structure.

In an application perspective, our work in this dissertation focuses on proposing novel community structure-based solutions for the following emerging problems: the forwarding and routing strategy in mobile networks, the worm containment problem in social networks and the limiting misinformation spread in online social networks. Our suggested strategies provide a significant improvement in terms of the solution quality for those mentioned problems, and promise a wider range of applications enabled by dynamic complex networks.

REFERENCES

[1] Ahn, Yong-Yeol, Bagrow, James P., and Lehmann, Sune. "Link communities reveal multi-scale complexity in networks." *Nature* 466 (2010): 761+.

URL http://arxiv.org/abs/0903.3178

[2] Andreev, K. and Räcke, H. "Balanced graph partitioning." *Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*. SPAA '04. New York, NY, USA: ACM, 2004, 120–124.

[3] Backstrom, Lars and Leskovec, Jure. "Supervised random walks: predicting and recommending links in social networks." *Proceedings of the fourth ACM international conference on Web search and data mining*. WSDM '11. New York, NY, USA: ACM, 2011, 635–644.

URL http://doi.acm.org/10.1145/1935826.1935914

[4] Barnes, Earl R. "An Algorithm for Partitioning the Nodes of a Graph." *SIAM Journal on Algebraic and Discrete Methods* 3 (1982).4: 541–550.

URL http://link.aip.org/link/?SML/3/541/1

[5] Berry, Michael W., Browne, Murray, Langville, Amy N., Pauca, V. Paul, and Plemmons, Robert J. "Algorithms and applications for approximate nonnegative matrix factorization." *Computational Statistics and Data Analysis*. 2006, 155–173.

[6] Blondel, Vincent D, Guillaume, Jean-Loup, Lambiotte, Renaud, and Lefebvre, Etienne. "Fast unfolding of communities in large networks." *Journal of Statistical Mechanics: Theory and Experiment* 2008 (2008).10: P10008.

URL http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

[7] Bose, Abhijit and Shin, Kang G. "Proactive security for mobile messaging networks." *Proceedings of the 5th ACM workshop on Wireless security*. WiSe '06. New York, NY, USA: ACM, 2006, 95–104.

URL http://doi.acm.org/10.1145/1161289.1161307

[8] Brandes, Ulrik, Delling, Daniel, Gaertler, Marco, Gorke, Robert, Hoefer, Martin, Nikoloski, Zoran, and Wagner, Dorothea. "On Modularity Clustering." *IEEE Trans. on Knowl. and Data Eng.* 20 (2008).2: 172–188.

URL http://dx.doi.org/10.1109/TKDE.2007.190689

[9] Cazabet, R., Amblard, F., and Hanachi, C. "Detection of Overlapping Communities in Dynamical Social Networks." *Social Computing (SocialCom), 2010 IEEE Second International Conference on*. 2010, 309 –314.

[10] Chaintreau, Augustin, Hui, Pan, Crowcroft, Jon, Diot, Christophe, Gass, Richard, and Scott, James. "Impact of Human Mobility on Opportunistic Forwarding Algorithms." *Mobile Computing, IEEE Transactions on* 6 (2007).6: 606 –620.

[11] Chen, Wei, Wang, Chi, and Wang, Yajun. "Scalable influence maximization for prevalent viral marketing in large-scale social networks." *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '10. New York, NY, USA: ACM, 2010, 1029–1038.

URL http://doi.acm.org/10.1145/1835804.1835934

[12] Chen, Wei, Yuan, Yifei, and Zhang, Li. "Scalable Influence Maximization in Social Networks under the Linear Threshold Model." *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM '10. Washington, DC, USA: IEEE Computer Society, 2010, 88–97.

URL http://dx.doi.org/10.1109/ICDM.2010.118

[13] Cichocki, A., Lee, H., Kim, Y-D, and Choi, S. "Non-negative matrix factorization with α-divergence." *Pattern Recognition Letters* ('08).

[14] Cichocki, A. and Zdunek, R. "Multilayer nonnegative matrix factorization using projected gradient approaches." *Proc. 13th International Conference on Neural Information Processing* ('07).

[15] Cichocki, A., Zdunek, R., and Amari, S.-i. "Nonnegative Matrix and Tensor Factorization [Lecture Notes]." *Signal Processing Magazine, IEEE* 25 (2008).1: 142 –145.

[16] Cichocki, Andrzej, Zdunek, Rafal, Phan, Anh Huy, and Amari, Shun-ichi. *Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation*. Wiley Publishing, 2009.

[17] Clauset, Aaron, Newman, M. E. J., and Moore, Cristopher. "Finding community structure in very large networks." *Physical Review E* 70 (2004).6: 066111+.

URL http://dx.doi.org/10.1103/PhysRevE.70.066111

[18] Cover, T. M. and Thomas, J. A. *Elements of Information Theory*. Wiley-Interscience, 1991.

[19] Daly, Elizabeth M. and Haahr, Mads. "Social network analysis for routing in disconnected delay-tolerant MANETs." *Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*. MobiHoc '07. New York, NY, USA: ACM, 2007, 32–40.

URL http://doi.acm.org/10.1145/1288107.1288113

[20] Danon, L., Diaz-Guilera, A., Duch, J., and Arenas, A. "Comparing community structure identification." *Journal of Statistical Mechanics: Theory and Experiment* 2005 (2005).09: P09008.

[21] Data, Foursquare. "sites.google.com/site/namnpuf/original_foursquare.7z." *Collected data*. 2012, 0–0.

[22] dataset, ArXiv. "http://www.cs.cornell.edu/projects/kddcup/datasets.html." *KDD Cup 2003* (2003).

[23] Delvenne, J-C C., Yaliraki, S. N., and Barahona, M. "Stability of graph communities across time scales." *Proceedings of the National Academy of Sciences of the United States of America* 107 (2010).29: 12755–12760.

URL http://dx.doi.org/10.1073/pnas.0903215107

[24] Ding, Chris, Li, Tao, and Peng, Wei. "On the equivalence between Non-negative Matrix Factorization and Probabilistic Latent Semantic Indexing." *Comput. Stat. Data Anal.* 52 (2008).8: 3913–3927.

URL http://dx.doi.org/10.1016/j.csda.2008.01.011

[25] Dinh, Thang N., Xuan, Ying, Thai, My T., Pardalos, Panos M., and Znati, Taieb. "On new approaches of assessing network vulnerability: hardness and approximation." *IEEE/ACM Trans. Netw.* 20 (2012).2: 609–619.

[26] Dinh, T.N., Xuan, Ying, and Thai, M.T. "Towards social-aware routing in dynamic communication networks." *Performance Computing and Communications Conference (IPCCC), 2009 IEEE 28th International*. 2009, 161 –168.

[27] Duan, Dongsheng, Li, Yuhua, Jin, Yanan, and Lu, Zhengding. "Community mining on dynamic weighted directed graphs." *Proceedings of the 1st ACM international workshop on Complex networks meet information & knowledge management*. CNIKM '09. New York, NY, USA: ACM, 2009, 11–18.

URL http://doi.acm.org/10.1145/1651274.1651278

[28] E, Weinan, Li, Tiejun, and Vanden-Eijnden, Eric. "Optimal partition and effective dynamics of complex networks." *Proceedings of the National Academy of Sciences of the United States of America* 105 (2008).23: 7907–7912.

URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=2786939&tool=pmcentrez&rendertype=abstract

[29] Eagle, Nathan and (Sandy) Pentland, Alex. "Reality mining: sensing complex social systems." *Personal Ubiquitous Comput.* 10 (2006).4: 255–268.

URL http://dx.doi.org/10.1007/s00779-005-0046-3

[30] Fire, Michael, Tenenboim, Lena, Lesser, Ofrit, Puzis, Rami, Rokach, Lior, and Elovici, Yuval. "Link Prediction in Social Networks Using Computationally Efficient Topological Features." *SocialCom/PASSAT*. IEEE, 2011, 73–80.

URL http://dblp.uni-trier.de/db/conf/socialcom/socialcom2011.html#FireTLPRE11

[31] Foobface. "facebook_virus_turns_your_computer_into_a_zombie.html, http://www.pcworld.com/article/155017/." *PC World*. 2008, 1.

[32] Fortunato, S. "Community detection in graphs." *Physics Reports* 486 (2010).3-5: 75 – 174.

[33] Fortunato, S. and Castellano, C. "Community Structure in Graphs." *eprint arXiv: 0712.2716* (2007).

[34] Fortunato, Santo. "Community detection in graphs." *Physics Reports* 486 (2010): 75–174.

[35] Fortunato, Santo and Barthelemy, Marc. "Resolution limit in community detection." *Proceedings of the National Academy of Sciences* 104 (2007).1: 36–41.

URL http://www.pnas.org/content/104/1/36.abstract

[36] Garey, Michael R. and Johnson, David S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[37] Girvan, M. and Newman, M. E. J. "Community structure in social and biological networks." *Proceedings of the National Academy of Sciences* 99 (2002).12: 7821–7826.

URL http://dx.doi.org/10.1073/pnas.122653799

[38] ———. "Community structure in social and biological networks." *PNAS* 99 (2002).

[39] Goldberg, M., Kelley, S., Magdon-Ismail, M., Mertsalov, K., and Wallace, A. "Finding Overlapping Communities in Social Networks." *Social Computing (SocialCom), 2010 IEEE Second International Conference on*. 2010, 104 –113.

[40] Gregory, Steve. "Finding overlapping communities in networks by label propagation." *New Journal of Physics* 12 (2010).10: 103018.

[41] Grubesic, T. H., Matisziw, T. C., Murray, A. T., and Snediker, D. "Comparative approaches for assessing network vulnerability." *Inter. Regional Sci. Review* 31 (2008).

[42] Guimera, Roger and Amaral, Luis A. Nunes. "Functional cartography of complex metabolic networks." *Nature* 433 (2005).7028: 895–900.

[43] Hoffmann, K H and Salamon, P. "Bounding the lumping error in Markov chain dynamics." *Applied Mathematics Letters* 22 (2009).9: 1471–1475.

URL http://www.google.com/search?client=safari&rls=en-us&q=Bounding+the+lumping+error+in+Markov+chain+dynamics&ie=UTF-8&oe=UTF-8

[44] Hopcroft, John, Khan, Omar, Kulis, Brian, and Selman, Bart. "Tracking evolving communities in large linked networks." *Proceedings of the National Academy of Sciences* 101 (2004): 5249–5253.

URL http://www.pnas.org/cgi/content/full/101/suppl_1/5249

[45] Hui, Pan, Crowcroft, J., and Yoneki, E. "BUBBLE Rap: Social-Based Forwarding in Delay-Tolerant Networks." *Mobile Computing, IEEE Transactions on* 10 (2011).11: 1576 –1589.

[46] Hui, Pan and Crowcroft, Jon. "How Small Labels Create Big Improvements." *Pervasive Computing and Communications Workshops, 2007. PerCom Workshops '07. Fifth Annual IEEE International Conference on*. 2007, 65 –70.

[47] Hui, Pan, Crowcroft, Jon, and Yoneki, Eiko. "Bubble rap: social-based forwarding in delay tolerant networks." *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing*. MobiHoc '08. New York, NY, USA: ACM, 2008, 241–250.

[48] Hui, Pan, Yoneki, Eiko, Chan, Shu Yan, and Crowcroft, Jon. "Distributed community detection in delay tolerant networks." *Proceedings of 2nd ACM/IEEE international workshop on Mobility in the evolving internet architecture*. MobiArch '07. New York, NY, USA: ACM, 2007, 7:1–7:8.

URL http://doi.acm.org/10.1145/1366919.1366929

[49] Jamali, Mohsen, Haffari, Gholamreza, and Ester, Martin. "Modeling the Temporal Dynamics of Social Rating Networks Using Bidirectional Effects of Social Relations and Rating Patterns." *Proceedings of the 2010 IEEE International Conference on Data Mining Workshops*. ICDMW '10. Washington, DC, USA: IEEE Computer Society, 2010, 344–351.

URL http://dx.doi.org/10.1109/ICDMW.2010.103

[50] Kemeny, John G., Hazleton, Mirkil, J. Laurie, Snell, and Gerald L., Thompson. "Finite Mathematical Structures." *1st edition. Englewood Cliffs, N.J.: Prentice-Hall, Inc* (1959).

[51] Kim, Hyang-Ah and Karp, Brad. "Autograph: toward automated, distributed worm signature detection." *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, 19–19.

URL http://dl.acm.org/citation.cfm?id=1251375.1251394

[52] Kim, Min-Soo and Han, Jiawei. "A particle-and-density based evolutionary clustering method for dynamic networks." *Proc. VLDB Endow.* 2 (2009).1: 622–633.

URL http://dl.acm.org/citation.cfm?id=1687627.1687698

[53] Koobface. "http://news.cnet.com/koobface-virus-hits-facebook/." *CNET*. 2008, 1.

[54] Kovacs, Istvan A., Palotai, Robin, Szalay, Mate S., and Csermely, Peter. "Community Landscapes: An Integrative Approach to Determine Overlapping Network Module Hierarchy, Identify Key Nodes and Predict Network Dynamics." *PLoS ONE* 5 (2010).9: e12528.

[55] Lancichinetti, A. and Fortunato, S. "Community detection algorithms: A comparative analysis." *Phys. Rev. E* 80 (2009).5: 056117.

[56] Lancichinetti, A., Fortunato, S., and Jnos, K. "Detecting the overlapping and hierarchical community structure in complex networks." *New Journal of Physics* 11 (2009).3: 033015.

[57] Lancichinetti, Andrea and Fortunato, Santo. "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities." *Phys. Rev. E* 80 (2009).1: 016118.

URL http://pre.aps.org/abstract/PRE/v80/i1/e016118

[58] ———. "Community detection algorithms: A comparative analysis." *Phys. Rev. E* 80 (2009): 056117.

URL http://link.aps.org/doi/10.1103/PhysRevE.80.056117

[59] ———. "Consensus clustering in complex networks." *Scientific Reports* 2 (2012).

URL http://dx.doi.org/10.1038/srep00336

[60] Lancichinetti, Andrea, Radicchi, Filippo, Ramasco, Jos J., and Fortunato, Santo. "Finding Statistically Significant Communities in Networks." *PLoS ONE* 6 (2011).4: e18961.

URL http://dx.doi.org/10.1371%2Fjournal.pone.0018961

[61] ———. "Finding Statistically Significant Communities in Networks." *PLoS ONE* 6 (2011).4: e18961.

URL http://dx.doi.org/10.1371%2Fjournal.pone.0018961

[62] Lazar, A., Abel, D., and Vicsek, T. "Modularity measure of networks with overlapping communities." *EPL (Europhysics Letters)* 90 (2010).1: 18001.

URL http://stacks.iop.org/0295-5075/90/i=1/a=18001

[63] Lee, C., Reid, F., McDaid, A., and Hurley, N. "Detecting highly overlapping community structure by greedy clique expansion." *Proceedings of the 4th Workshop on Social Network Mining and Analysis* (2010).

[64] Lee, Daniel D. and Seung, H. Sebastian. "Algorithms for Non-negative Matrix Factorization." *In NIPS*. MIT Press, 2000, 556–562.

[65] Leskovec, Jure, Huttenlocher, Daniel, and Kleinberg, Jon. "Predicting positive and negative links in online social networks." *Proceedings of the 19th international conference on World wide web*. WWW '10. New York, NY, USA: ACM, 2010, 641–650.

URL http://doi.acm.org/10.1145/1772690.1772756

[66] Leskovec, Jure, Lang, Kevin J., Dasgupta, Anirban, and Mahoney, Michael W. "Statistical properties of community structure in large social and information networks." *Proceedings of the 17th international conference on World Wide Web*. WWW '08. New York, NY, USA: ACM, 2008, 695–704.

URL http://doi.acm.org/10.1145/1367497.1367591

[67] Li, Tao and Ding, Chris. "The Relationships Among Various Nonnegative Matrix Factorization Methods for Clustering." *Proceedings of the Sixth International Conference on Data Mining*. ICDM '06. Washington, DC, USA: IEEE Computer Society, 2006, 362–371.

URL http://dx.doi.org/10.1109/ICDM.2006.160

[68] Li, Yanhua, Zhang, Zhi-Li, and Bao, Jie. "Mutual or Unrequited Love: Identifying Stable Clusters in Social Networks with Uni- and Bi-directional Links." *Algorithms and Models for the Web Graph*. eds. Anthony Bonato and Jeannette Janssen, vol. 7323 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012. 113–125.

URL http://dx.doi.org/10.1007/978-3-642-30541-2_9

[69] Liben-Nowell, David and Kleinberg, Jon. "The link prediction problem for social networks." *Proceedings of the twelfth international conference on Information and knowledge management*. CIKM '03. New York, NY, USA: ACM, 2003, 556–559.

URL http://doi.acm.org/10.1145/956863.956972

[70] Lin, Yu-Ru, Chi, Yun, Zhu, Shenghuo, Sundaram, Hari, and Tseng, Belle L. "Facetnet: a framework for analyzing communities and their evolutions in dynamic networks." *Proceedings of the 17th international conference on World Wide Web*. WWW '08. New York, NY, USA: ACM, 2008, 685–694.

URL http://doi.acm.org/10.1145/1367497.1367590

[71] ———. "Analyzing communities and their evolutions in dynamic social networks." *ACM Trans. Knowl. Discov. Data* 3 (2009).2: 8:1–8:31.

URL http://doi.acm.org/10.1145/1514888.1514891

[72] Lin, Yu-Ru, Sun, Jimeng, Castro, Paul, Konuru, Ravi, Sundaram, Hari, and Kelliher, Aisling. "MetaFac: community discovery via relational hypergraph factorization." *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. New York, NY, USA: ACM, 2009, 527–536.

URL http://doi.acm.org/10.1145/1557019.1557080

[73] Luxburg, Ulrike. "A tutorial on spectral clustering." *Statistics and Computing* 17 (2007).4: 395–416.

URL http://dx.doi.org/10.1007/s11222-007-9033-z

[74] Matisziw, Timothy C. and Murray, Alan T. "Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure." *Comput. Oper. Res.* 36 (2009).1: 16–26.

[75] Newman, M E J. "The Structure and Function of Complex Networks." *SIAM Review* 45 (2003).2: 167–256.

URL http://link.aip.org/link/SIREAD/v45/i2/p167/s1&Agg=doi

[76] Newman, M. E. J. "Fast algorithm for detecting community structure in networks." *Phys. Rev. E* 69 (2004): 066133.

URL http://link.aps.org/doi/10.1103/PhysRevE.69.066133

[77] ———. "Finding community structure in networks using the eigenvectors of matrices." *Physical Review E* 74 (2006).3: 036104.

[78] Newman, M. E. J. and Girvan, M. "Finding and evaluating community structure in networks." *Physical Review* E 69 (2004).026113.

[79] Newman, M E J and Leicht, E A. "Mixture models and exploratory analysis in networks." *Proceedings of the National Academy of Sciences of the United States of America* 104 (2007).23: 9564–9569.

URL http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=1887592&tool=pmcentrez&rendertype=abstract

[80] Nguyen, Nam P., Dinh, Thang N., Tokala, Sindhura, and Thai, My T. "Overlapping communities in dynamic networks: their detection and mobile applications."

Proceedings of the 17th annual international conference on Mobile computing and networking. MobiCom '11. New York, NY, USA: ACM, 2011, 85–96.

URL http://doi.acm.org/10.1145/2030613.2030624

[81] Nguyen, N.P., Dinh, T.N., Xuan, Ying, and Thai, M.T. "Adaptive algorithms for detecting community structure in dynamic social networks." *INFOCOM, 2011 Proceedings IEEE*. 2011, 2282 –2290.

[82] Nguyen, N.P., Xuan, Ying, and Thai, M.T. "A novel method for worm containment on dynamic social networks." *MILITARY COMMUNICATIONS CONFERENCE, 2010 - MILCOM 2010*. 2010, 2180 –2185.

[83] Nicosia, V., Mangioni, G., Carchiolo, V., and Malgeri, M. "Extending the definition of modularity to directed graphs with overlapping communities." *J. Stat. Mech.: Theory and Experiment* 2009 (2009).03: P03024.

[84] Palla, G., Derenyi, I., Farkas1, I., and Vicsek, T. "Uncovering the overlapping community structure of complex networks in nature and society." *Nature* 435 (2005).10.

[85] Palla, G., Pollner, P., Barabasi, A., and Vicsek, T. "Social Group Dynamics in Networks." *Adaptive Networks* (2009).

[86] Palla, Gergely, Barabasi, Albert-Laszlo, and Vicsek, Tamas. "Quantifying social group evolution." *Nature* 446 (2007).7136: 664–667.

URL http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?cmd=Retrieve\&db=pubmed\&dopt=Abstract\&list_uids=17410175

[87] Panzarasa, P., Opsahl, T., and Carley, K. M. "Patterns and dynamics of users' behavior and interaction: Network analysis of an online community." *J. American Soc. of Info. Sci. Tech. 60(5)* ('09).

[88] Peters, Karsten, Buzna, Lubos, and Helbing, Dirk. "Modelling of cascading effects and efficient response to disaster spreading in complex networks." *IJCIS* 4 (2008).1/2: 46–62.

URL http://dblp.uni-trier.de/db/journals/ijcritis/ijcritis4.html#PetersBH08

[89] Piccardi, Carlo. "Finding and Testing Network Communities by Lumped Markov Chains." *PLoS ONE* 6 (2011).11: e27028.

URL http://dx.doi.org/10.1371%2Fjournal.pone.0027028

[90] Psorakis, I., Roberts, S., and Ebden, M. "Overlapping community detection using Bayesian non-negative matrix factorization." *Phys. Rev. E. 83* (11).

[91] Radicchi, Filippo, Castellano, Claudio, Cecconi, Federico, Loreto, Vittorio, and Parisi, Domenico. "Defining and identifying communities in networks." *Proceedings of the National Academy of Sciences of the United States of America* 101 (2004).9: 2658–2663.

URL http://www.pnas.org/content/101/9/2658.abstract

[92] Reichardt, Joerg and Bornholdt, Stefan. "Detecting fuzzy community structures in complex networks with a Potts model." *Physical Review Letters* 93 (2004).21: 218701.

URL http://arxiv.org/abs/cond-mat/0402349

[93] Rosvall, Martin and Bergstrom, Carl T. "Mapping Change in Large Networks." *PLoS ONE* 5 (2010).1: e8694.

URL http://dx.doi.org/10.1371%2Fjournal.pone.0008694

[94] Scott, James, Gass, Richard, Crowcroft, Jon, Hui, Pan, Diot, Christophe, and Chaintreau, Augustin. "CRAWDAD trace cambridge/haggle/imote/infocom2006 (v. 2009-05-29)." Downloaded from http://crawdad.cs.dartmouth.edu/cambridge/haggle /imote/infocom2006, 2009.

[95] Scripps, Jerry, Tan, Pang-Ning, and Esfahanian, Abdol-Hossein. "Node roles and community structure in networks." *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*. WebKDD/SNA-KDD '07. New York, NY, USA: ACM, 2007, 26–35.

[96] Sean P., Meyn and Richard L., Tweedie. "Markov Chains and Stochastic Stability." *2nd edition. Cambridge University Press* (2009).

[97] Sekar, Vyas, Xie, Yinglian, Reiter, Michael K., and Zhang, Hui. "A Multi-Resolution Approach forWorm Detection and Containment." *Proceedings of the International Conference on Dependable Systems and Networks*. DSN '06. Washington, DC, USA: IEEE Computer Society, 2006, 189–198.

URL http://dx.doi.org/10.1109/DSN.2006.6

[98] Sun, Jimeng, Faloutsos, Christos, Papadimitriou, Spiros, and Yu, Philip S. "GraphScope: parameter-free mining of large time-evolving graphs." *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '07. New York, NY, USA: ACM, 2007, 687–696.

URL http://doi.acm.org/10.1145/1281192.1281266

[99] Tantipathananandh, Chayant and Berger-Wolf, Tanya. "Constant-factor approximation algorithms for identifying dynamic communities." *Proceedings*

of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. KDD '09. New York, NY, USA: ACM, 2009, 827–836.

URL http://doi.acm.org/10.1145/1557019.1557110

[100] Viswanath, Bimal, Mislove, Alan, Cha, Meeyoung, and Gummadi, Krishna P. "On the evolution of user interaction in Facebook." *Proceedings of the 2nd ACM workshop on Online social networks*. WOSN '09. New York, NY, USA: ACM, 2009, 37–42.

URL http://doi.acm.org/10.1145/1592665.1592675

[101] Šíma, Jiří and Schaeffer, Satu Elisa. "On the NP-Completeness of some graph cluster measures." *Proceedings of the 32nd conference on Current Trends in Theory and Practice of Computer Science*. SOFSEM'06. Berlin, Heidelberg: Springer-Verlag, 2006, 530–537.

URL http://dx.doi.org/10.1007/11611257_51

[102] Wang, Fei, Li, Tao, Wang, Xin, Zhu, Shenghuo, and Ding, Chris. "Community discovery using nonnegative matrix factorization." *Data Min. Knowl. Discov.* 22 (2011).3: 493–521.

URL http://dx.doi.org/10.1007/s10618-010-0181-y

[103] ———. "Community discovery using nonnegative matrix factorization." *Data Min. Knowl. Discov.* 22 (2011).3: 493–521.

URL http://dx.doi.org/10.1007/s10618-010-0181-y

[104] Wang, Pu, González, Marta C., Hidalgo, Cesar A., and Barabasi, Albert-Laszlo. "Understanding the Spreading Patterns of Mobile Phone Viruses." *Science* 324 (2009).5930: 1071–1076.

URL http://www.barabasilab.com/pubs/CCNR-ALB_Publications/200904-02_ScienceExpr-PhoneViruses/200904-02_ScienceExpr-PhoneViruses

[105] Wang, Yang, Di, Zengru, and Fan, Ying. "Identifying and Characterizing Nodes Important to Community Structure Using the Spectrum of the Graph." *PLoS ONE* 6 (2011).11: e27418.

[106] Weaver, Nicholas, Staniford, Stuart, and Paxson, Vern. "Very fast containment of scanning worms." *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, 3–3.

URL http://dl.acm.org/citation.cfm?id=1251375.1251378

[107] Ye, Zhenqing, Hu, Songnian, and Yu, Jun. "Adaptive clustering algorithm for community detection in complex networks." *Phys. Rev. E* 78 (2008): 046115.

URL http://link.aps.org/doi/10.1103/PhysRevE.78.046115

[108] Zdunek, Rafal and Cichocki, Andrzej. "Non-negative matrix factorization with quasi-newton optimization." *Proceedings of the 8th international conference on Artificial Intelligence and Soft Computing*. ICAISC'06. Berlin, Heidelberg: Springer-Verlag, 2006, 870–879.

URL http://dx.doi.org/10.1007/11785231_91

[109] Zhang, Yuzhou, Wang, Jianyong, Wang, Yi, and Zhou, Lizhu. "Parallel community detection on large networks with propinquity dynamics." *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '09. New York, NY, USA: ACM, 2009, 997–1006.

URL http://doi.acm.org/10.1145/1557019.1557127

[110] Zhu, Zhichao, Cao, Guohong, Zhu, Sencun, Ranjan, S., and Nucci, A. "A Social Network Based Patching Scheme for Worm Containment in Cellular Networks." *INFOCOM 2009, IEEE*. 2009, 1476 –1484.

BIOGRAPHICAL SKETCH

Nam P. Nguyen is currently at his fourth year PhD student in Department of Computer and Information Science and Engineering (CISE), University of Florida and a member of Optima Network Science Lab under the guidance of Professor My T. Thai. Prior to his Ph.D. study, Nam received my B.S. and M.S. degrees both in applied mathematics from Vietnam National University (2007) and Ohio University (2009).

His research interests include dynamic complex network problems, such as non-overlapping and overlapping network community structure, worm and virus containment, social networks; cascading failures; combinatorial optimization and approximation algorithms. In particular, his current research focuses on designing adaptive algorithms for effectively identify communities in dynamic networks such as mobile or online social networks, as well as their applications in different aspects of networking problems. In addition, he is also interested in effective methods to stop the propagation of virus, worm and misinformation spread on large scale dynamic networks, in terms of both approximation and social-based algorithms.

During his Ph.D. study, Nam has published many papers in top-tier conferences and journals including INFOCOM, MOBICOM, WEBSCI and IEEE Transaction on Mobile Computing, IEEE Transaction on Networking, etc. In 2011, Nam spent his summer as an intern in CCS-3 division, Los Alamos National Laboratories, where he conducted research and published a paper on containing the spread of misinformation in large scale online social networks. Nam also the recipient of many awards such as the Student Travel Grants of MILCOM'10, MOBICOM'11 and SIGWEB'12 conferences, Travel Grant of The College of the Engineering in 2011, University of Florida.

COMPLEX NETWORKS UNDER ATTACKS: VULNERABILITY ASSESSMENT AND OPTIMIZATION

By

THANG N. DINH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

I dedicate this disseration to my parents, Kien Duong and Thuy Dinh, and to my wife, Phuong-Thao, and my daughter, Linh-Thu.

ACKNOWLEDGMENTS

This dissertation would not have been possible without the help, support, guidance and efforts of so many people in so many ways. Firstly, I would like to thank my mentor Dr. My T. Thai for her invaluable support throughout my Ph.D program. I would always treasure her infectious passion, preciseness, profound knowledge, and valuable advice on becoming a good scientist and, more importantly, a better person.

I am also very grateful for my committee members, Dr. Ahmed Helmy, Dr. Tamer Kahveci, Dr. Panagote Pardalos, and Dr. Sartaj Sahni for the support theyve lent me over all these years as well as their valuable support for my future career. I would like also to thank Dr. Meera Sitharam for helping me with my questions and many insightful discussions. I would like to acknowledge and thank my friends Ravi Tiwari, Incheol Shin, Ying Xuan, Nam Nguyen, Dzung Nguyen, Yilin Shen, Ferhat Ay for their help in research and all the fun we had together. Finally I would like to take the opportunity to thank all my teachers who has equipped me with the requisite knowledge and skills. My research was partially supported by the DTRA YIP grant number HDTRA1-09-1-0061 and the NSF CAREER Award number 0953284.

TABLE OF CONTENTS

page

LIST OF TABLES

Table page

LIST OF FIGURES

Abstract of Proposal Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

COMPLEX NETWORKS UNDER ATTACKS:
VULNERABILITY ASSESSMENT AND OPTIMIZATION

By

Thang N. Dinh

Mar 2013

Chair: Professor My T. Thai
Major: Computer Engineering

Complex network systems are extremely vulnerable to attacks. In the presence of uncertainty, assessing *network vulnerability* before potential malicious attacks is vital for network planning and risk management.

In this dissertation, we apply optimization theory and approximation techniques to address the following fundamental questions: How do we quantitatively measure the vulnerability degree of the network? How to identify critical infrastructures in the network in the context of both *individual failures* and/or *cascading failures* that spread from one node to another across the network structure? The dissertation provides several *new theoretical frameworks* and *approximation algorithms* to characterize the network vulnerability and critical infrastructures, which advances the understanding of network vulnerability. The dissertation tackles the above questions by crossing and contributing new techniques to several research areas such as graph theory, approximation algorithms, mathematical programming, and computational complexity.

This research can potentially impact many applications that benefit from networks such as the Internet, smart grids, and transportation networks where vulnerability is an important characteristic.

CHAPTER 1
INTRODUCTION

Assessing network vulnerability before potential disruptive events such as natural disasters or malicious attacks is vital for network planning and risk management. It enables us to seek and safeguard against most destructive scenarios in which the overall network connectivity falls dramatically.

There have been numerous efforts on proposing evaluation measures of the network vulnerability, as summarized in [46]. On one hand, several global graph measures, such as Cyclomatic number, Maximum network circuits, Alpha index, and Beta index, which investigate basic graph properties, i.e., number of vertices, edges and pairwise shortest paths, are adopted to evaluate the network vulnerability. However, these global measures can neither be rigorously mapped to the over network connectivity, nor reveal the set of most critical vertices and edges, thus are not suitable to assess the network vulnerability in terms of connectivity. On the other hand, researchers focused on local nodal centrality [18], such as degree centrality, betweenness centrality and closeness centrality, in order to differentiate the critical vertices from the others, and further evaluate the network by quantifying such vertices. Unfortunately, being unable to cast these local properties to global network connectivity, these measures fail to indicate accurate vulnerabilities and cannot reveal the global damage done on the network under attacks.

To this end, in the first part of this proposal, we investigate a measure called *pairwise connectivity* and formulate this vulnerability assessment problem as a new graph-theoretical optimization problems. The pairwise connectivity is the sum of every vertex pair connectivity, which is quantified as 1 if they are (strongly) connected and 0 if not. Our new optimization problems, called β*-vertex disruptor* and β*-edge disruptor*, aim to discover the set of *critical* node/edges, whose removal results in the sharpest decline of the pairwise connectivity. With respect to a level of connectivity disruption, the more

vertices/edges required to be removed, the less vulnerable the network is; conversely, the fewer vertices/edges needed to removed, the easier this network is to be destroyed. The β-disruptor problems are defined in section 1.1.

The second part of this proposal focuses on assessing network vulnerability against cascading-failures, that spread among nodes of a power grid or communication network during a widespread outage, among financial institutions during a financial crisis, or through a human population during the outbreak of an epidemic disease. We develop a new measurement for cascading-resilience in networks subject to such cascading failures. The cascading-resilience (a.k.a. network vulnerability) is measured as the *minimum size of a set of nodes* that can trigger an *outbreak of failure to the whole network* in *a short amount of time*. Thus, we formulate the measuring cascading-resilience as an optimization problem, called cost-effective massive outbreak problem (CFM).

Since all formulated optimization are shown to be NP-complete, efficient algorithms to find the exact solutions for the formulated problems are unlikely to exist. Thus, we focus on designing algorithms that can provide guarantee on their performances, which are known as *approximation algorithms*. Furthermore, we also devote one part of the proposal to design scalable algorithms for large-scale networks, which have hundreds of millions links. Those algorithms are essential to benefit the available of big data.

1.1 Connectivity-based Vulnerability Assessment

1.1.1 Motivation

Connectivity plays a vital role in network performance and is fundamental to vulnerability assessment. Potential disruptive events, such as natural disasters or malicious attacks, which always destroy a set of interacting elements or connections, can dramatically compromise the connectivity and result in considerate decline of the network QoS, or even breakdown the whole network [24, 26, 46, 62, 63, 68]. Of this concern, pre-active evaluation over the network vulnerability with respect to connectivity, in order to defense such potential disruptions, is quite essential and beneficial to the

design and maintenance of any infrastructure networks, for example, communication, commercial, and social networks.

1.1.2 β-disruptor Problems

Besides the homogeneous network model consisting of uniform nodes and bidirectional links, the heterogeneous network model, where various interacting elements of different kinds are connected through unidirectional links with non-uniform expenses, finds numerous applications nowadays [53, 58, 72], but as well, exhibits multiple difficulties for optimization and maintenance. In the light of this, we abstract our general network model as a directed graph $G(V, E)$, where V refers to a set of nodes and E refers to a set of unidirectional links. The expense of each directed edge (u, v) between vertex u and v is quantified as a nonnegative value $c(u, v)$, for all the $m = |E|$ links among $n = |V|$ nodes.

As mentioned above, our evaluation over the network vulnerability is based on the value of *overall pairwise connectivity* in the abstracted graph, which is defined as follows: given any vertex pair $(u, v) \in V \times V$ in the graph, we say that they are connected iff there exists paths between u and v in *both directions* in G, i.e., strongly connected to each other. The pairwise connectivity $p(u, v)$ is quantified as 1 if this pair is connected, 0 otherwise. Since the main purpose of network lies in connecting all the interacting elements, we study on the aggregate pairwise connectivity between all pairs, that is, the sum of quantified pairwise connectivity, which we denote as $\mathcal{P}(G) = \sum_{u,v \in V \times V} p(u, v)$ for graph G. Apparently $\mathcal{P}(G)$ is maximized at $\binom{n}{2}$ when G is a strongly connected graph. Based on this, we have:

Definition 1. *(Edge disruptor) Given $0 \leq \beta \leq 1$, a subset $S \subset E$ in $G = (V, E)$ is a β-**edge disruptor** if the overall pairwise connectivity in the $G[E \setminus S]$, obtained by removing S from G, is no more than $\beta\binom{n}{2}$. By minimizing the cost of such edges in S, we have the β-**edge disruptor problem**, i.e., find a minimized β-edge disruptor in a strongly connected graph $G(V, E)$.*

Recall that G is strongly connected iff for every vertex v in G, there is a directed path from v to all other vertices. A subgraph of G is called a *strongly connected component* (SCC) iff it is a maximal subgraph of G with all vertex pairs u, v within it connected by directed paths in both directions. Assume that a β-edge disruptor disrupts the connectivity in $G(V, E)$ by separating it into several smaller SCCs, say C_i for $i = 1 \dots l$ i.e. $V = \biguplus_{i=1}^{l} C_i$. We have:

$$\begin{aligned} \mathcal{P}(G) &= \sum_{i=1}^{l} \binom{|C_i|}{2} = \frac{1}{2}\left(\sum_{i=1}^{l} |C_i|^2 - |V|\right) \\ &= \frac{1}{2}\left(\frac{n^2}{l} - n\right) + \frac{l}{2} Var(C) \end{aligned}$$

where $Var(C) = \frac{1}{l}\sum_{i=1}^{l}(|C_i| - \overline{C})^2 = \frac{1}{l}\sum_{i=1}^{l}(|C_i| - \frac{n}{l})^2$. Therefore, the two key factors affecting pairwise connectivity are the number of SCCs and the variance of their sizes. They provide us an alternative measure for evaluating the structural balance and fragmentation of the network.

Similarly, we define β-*vertex disruptor* and its corresponding optimization problem:

β-vertex disruptor problem: Given a strongly connected graph $G(V, E)$ and a fixed number $0 \leq \beta \leq 1$, find a subset $S \subseteq V$ with the minimum size such that the total pairwise connectivity in $G_{[V \setminus S]}$, obtained by removing S from G, is no more than $\beta\binom{n}{2}$. Such a set S is called β-vertex disruptor.

1.1.3 Related Work

The classic vulnerability measurements are mainly based on the centrality of each vertex in the graph, which consist of degree centrality, betweenness, closeness, and eigenvector centrality [18]. However, these measures fail to indicate accurate vulnerabilities and cannot reveal the global damage done on the network under attacks.

On the other hand, the global graph measures are mainly functions of graph properties, e.g., the number of vertices/edges, operational O-D pairs, operational

paths, minimum shortest paths [24, 46, 62]. However, some of these attributes cannot be calculated in polynomial-time for dense networks. In essence, these functions do not reveal the set of most critical vertices and edges, thus are not suitable to assess the network vulnerability in terms of connectivity. Several similar concepts with our pairwise connectivity have been recently investigated in [10, 17, 73], where the terms *average pairwise connectivity*, *pairwise connected ratio* and *cohesion* were used. However, none of them were able to formulate the calculation of this measure as an optimization problem and provide the hardness proof along with performance guaranteed approximation algorithms. Moreover, the problem β-disruptor studied in this paper take into account the roles of all edges and vertices in the global network connectivity, thus provides a more essential research and thorough analysis over the underlying vulnerability framework established.

As a subproblem of this vulnerability assessment problem, Critical Vertex/Edge, which are defined as the minimum number of vertices/edges whose removal disconnects the graph, are also studied and solved using extensive heuristics, however, without performance guarantee. Some work of this kind in the context of wireless network are [44][47][48], nevertheless, these works consider only whether or not the graph is disconnected and ignore how fragmental the graph becomes. They are insufficient to evaluate the graph vulnerability.

Bissias et al. [14, 15] study the problem of bounding the damage under link attacks. However, the provided methods either require solving costly semidefinite programming problem [15] or involving weak bounds due to the presence of partitions with negative sizes [14].

1.1.4 Contributions

Our contributions for the vulnerability assessment research are as follows:

- Providing a novel underlying framework toward the vulnerability assessment by investigating the pairwise connectivity and formulating it as an optimization

problem β-*disruptor* on general graphs, which consists of two versions β-*vertex disruptor* and β-*edge disruptor*;

- Proving the NP-completeness of the two problems above and further proving that no PTAS exists for β-*vertex disruptor*;

- Presenting an $O(\log^{\frac{3}{2}} n)$ pseudo-approximation algorithm for β-*vertex disruptor*, and an $O(\log n \log\log n)$ pseudo-approximation algorithm for β-*edge disruptor*. These solutions can be applied to both homogeneous networks and heterogeneous networks with unidirectional links and non-uniform nodal properties.

- We present a spectral lower-bound method for the link vulnerability assessment problem, β-edge disruptor. The new lower-bound method is useful in both comparing the vulnerability of different networks and providing guarantees for other heuristics assessment methods.

- In Chapter 4, we present an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. Since β-vertex disruptor is a special case of β-disruptor, the algorithm implies an $O(\sqrt{\log n})$ bicriteria approximation algorithm for β-vertex disruptor, which improve the best result for β-vertex disruptor, the $O(\log n \log\log n)$ bicriteria approximation algorithm.

- In probabilistic networks, We first show that computing expected pairwise connectivity is #P-complete. In addition, we develop a Fully Polynomial Time Approximation Scheme (FPRAS) to estimate network connectivity with an arbitrary precision.

1.2 Cascading Failures in Critical Infrastructures

Malicious attacks can cause failures to spread over the network. Such cascading processes can be found in contagious failures that spread among nodes of a power grid or communication network during a widespread outage, among financial institutions during a financial crisis, or through a human population during the outbreak of an epidemic disease. During the cascade process, nodes are assigned states which change because of the influence of their neighbors. For example, an infected node can pass the infection to its contacts in the network, and the infection could then be passed to more and more nodes. We focus on the case where nodes change their state only when a certain fraction of their neighbors exert influence (see e.g. [21, 81, 82]).

We develop a new measurement for cascading-resilience in networks subject to such cascading failures. The cascading-resilience (a.k.a. network vulnerability) is measured as the *minimum size of a set of nodes* that can trigger an *outbreak of failure to the whole network* in *a short amount of time*. Thus, we formulate the measuring cascading-resilience as an optimization problem, called cost-effective massive outbreak problem (CFM). The key difference in comparison with other works on cascading failure and diffusion process is that we consider the time-aspect of the outbreak. We limit the propagation of failure to within d hops from the failure sources.

Both analytical analysis based on scale-free network theory and numerical analysis demonstrate that the massive outbreak might involve costly seeding. To minimize the seeding cost, we provide mathematical programming to find optimal seeding for medium-size networks and propose VirAds, an efficient algorithm, to tackle the problem on large-scale networks. VirAds guarantees a relative error bound of $O(1)$ from the optimal solutions in power-law networks and outperforms the greedy heuristics which realizes on the degree centrality. Moreover, we also show that, in general, approximating the optimal seeding within a ratio better than $O(\log n)$ is unlikely possible.

1.2.1 Problem Definitions

We are given a network modeled as an undirected graph $G = (V, E)$ where the vertices in V represent users in the network and the edges in E represent social links between users. We use n and m to denote the number of vertices and edges, respectively. The set of neighbors of a vertex $v \in V$ is denoted by $N(v)$ and we denote by $d(v) = |N(v)|$ the degree of node v.

We continue with specifying the diffusion model that governs the process of cascading failures. Existing diffusion models can be categorized into two main groups [49]:

- *Threshold model.* Each node v in the network has a threshold $t_v \in [0, 1]$, typically drawn from some probability distribution. Each connection (u, v) between nodes u

and v is assigned a weight $w(u, v)$. For a node v, let $F(v)$ be the set of neighbors of v that are already influenced. Then v is influenced if $t_v \leq \sum_{u \in F(v)} w(u, v)$.

- *Cascade model*. Whenever a node u is influenced, it is given a single chance to activate each of its neighbor v with a given probability $p(u, v)$.

Most papers on cascading processes assume that the probabilities $p(u, v)$ or weights $w(u, v)$ and thresholds t_v are given as a part of the input. However, they are generally not available and inferring those probabilities and thresholds has remained a non trivial problem [43]. Therefore, in addition to the bounded propagation hop, we use a simplified variation of the linear threshold model in which a vertex is activated if a fraction ρ of its neighbors are active as follows.

Locally Bounded Diffusion Model. Let $R_0 \subset V$ be the subset of vertices selected to initiate the influence propagation, which we call the *seeding*. We also call a vertex $v \in R_0$ a seed. The propagation process happens in round, with all vertices in R_0 are influenced (thus active in adopting the behavior) at round $t = 0$. At a particular round $t \geq 0$, each vertex is either active (adopted the behavior) or inactive and each vertex's tendency to become active increases when more of its neighbors become active. If an inactive vertex u has more than $\lceil \rho \, d(u) \rceil$ active neighbors at round t, then it becomes active at round $t + 1$, where ρ is the *influence factor* as discussed later. The process goes on for a maximum number of d rounds and a vertex once becomes active will remain active until the end. We say an initial set R_0 of vertices to be a d-*seeding* if R_0 can make all vertices in the networks active within at most d rounds.

The influence factor $0 < \rho < 1$ is a constant that decides how widely and quickly the influence propagates through the network. Influence factor ρ reflects real-world factors such as how easy to share the content with others, or some intrinsic benefit for those who initially adopt the behavior. In case $\rho = 1/2$ the model is also known as the *majority* model that has many application in distributed computing, voting system [66], etc.

Problem Definition. Given the diffusion model, the *Cost-effective, Fast, and Massive outbreak (CFM)* problem is defined as follows

Definition 2 (CFM Problem)**.** *Given an undirected graph* $G = (V, E)$ *modeling a complex network and an influence factor* $0 < \rho < 1$*, find in* V *a minimum size* d*-seeding i.e. a subset of vertices that can activate all vertices in the network within at most* d *rounds.*

Generalization. The diffusion model can be generalized in several ways. For example, the model can be extended naturally to cover directed networks or specify different influence factor ρ_v for each node $v \in V$. For simplicity we stick with the current model to avoid setting parameters during the experiments. Nevertheless, major results such as the approximation ratio of the VirAds algorithm or the hardness of approximation results still hold for the generalized models.

1.2.2 Related Work.

Outbreak can be thought of as a diffusion of information about the product and its adoption over the network. Kempe et al. [49, 50] formulated the influence maximization problem as an optimization problem. They showed the problem to be NP-complete and devised an $(1 - 1/e - \epsilon)$ approximation algorithm. A major drawback of their algorithm is that the accuracy ϵ, and efficiency depends on the number of times running Monte-Carlo simulation of the propagation model. Later, Leskovec et al. [55] study the influence propagation in a different perspective in which they aim to find a set of nodes in networks to detect the spread of virus as soon as possible. They improve the simple greedy method to run faster. The greedy algorithm is furthered improved by Chen et al. [22] by using an influence estimation. However, the proposed algorithm might only perform well for small values of propagation probabilities. In addition, the algorithm time complexity should be $O((m + k) \log n)$ instead of the claimed $O(k \log m + m)$.

Influence propagation with limited number of hops is first considered in Wang *et al.* [78, 83] in which the proposed heuristic has high time complexity. Feng et al. [82] show NP-completeness for the problem. We note that none of the mentioned approaches handled large-scale social networks of million of nodes as we shall study in Section 6.4.

1.2.3 Contributions

Our contributions are summarized as follows:

- Our first finding shows that the seeding for fast and massive spreading must contain a non-trivial fraction of nodes in the networks, which is cost-prohibitive for large-scale networks. This is confirmed by both our theoretical analysis based on the power-law model in [4] and our extensive experiments.
- We propose VirAds, a scalable algorithm to find a set of minimal seeding to expeditiously propagate the influence to the whole network. VirAds outperforms the greedy heuristics based on well-known degree centrality and scales up to networks of hundred of million links. We prove that the algorithm guarantees a relative error bound of $O(1)$, assuming that the network is power-law.
- We show how hard to obtain a near optimal solution for CFM by proving the impossibility to approximate the optimal solution within a ratio better than $O(\log n)$.

CHAPTER 2
MULTIPLE LINK ATTACKS

We convert the vulnerability assessment into a graph-theoretical optimization problem: *finding a minimized set of vertices/edges whose removal degrades the pairwise connectivity to a desired degree*. Considering that disrupting these vertices and edges will considerately degrade the network performance, we refer to them as β-*disruptor* throughout this paper, where $0 \leq \beta < 1$ denotes the fraction of desired pairwise connectivity (which we will define later). Two new optimization problems β-*vertex disruptor* and β-*edge disruptor* will be studied and proved to be NP-complete. We addressed them with several pseudo-approximation algorithms with provable performance bounds, which thus ensure the feasibility and accuracy of this evaluation measure.

The benefit of our new measure can be briefly illustrated in Fig.2-1, compared with the assessment using degree centrality. Notice that both networks A and B have 7 vertices and are originally strongly connected. According to the nodal degree centrality, removing the black vertex with maximum outgoing degree 5 in Fig.2-1-(a) leaves the network A still strongly connected with 5 vertices; and removing the black vertex with maximum outgoing degree 4 in Fig.2-1-(b) partitions the graph into two strongly connected components. In this sense, network A is somewhat stronger (less vulnerable) than B. However, our model can discover that, deleting only the grey vertex in A will be enough to decrease the overall connectivity to 0; on the contrary, at least 3 vertices in B are required to be removed to make overall connectivity 0. Therefore, A is actually much more vulnerable. Apparently, our measure provides more accurate assessment.

Furthermore, our study over the multiple disruption levels (different values of β) presents a deeper meaning and greater potentials. Several recent studies in the context of wireless networks have aimed to discover the nodes/edges whose removal disconnects the network, regardless of how disconnected it is [44][47][48]. Apparently,

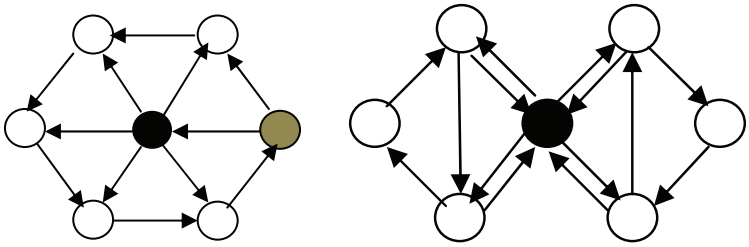


Figure 2-1. After the “central” vertex (in black) with maximum out-going degree is removed, network (A) is still strongly connected while (B) is fragmented; however in fact, only removing one vertex (in grey) is enough to destroy network (A).

this is a weaker version of our β-*disruptor*, since no specification over the quantified network connectivity is concerned. However, it is not reasonable to limit the possible disruption to only disconnecting the graph, ignoring how fragmented it is. For instance, a scale-free network can tolerate high random failure rates [12], since the destruction to boundary vertices may not significantly decline the network connectivity even though the whole graph becomes disconnected. In addition, different disruption levels may require different sets of disruptor on which our model can differentiate whereas existing methods cannot. For example, the node centrality method always returns a set of nodes with non-increasing degrees regardless of the disruption level.

This chapter is organized as follows. We first provide the hardness results in Section 6.3. The pseudo-approximation algorithms for β-*edge disruptor* and β-*vertex disruptor* are presented in Section 2.2 and Section 3.1 respectively. We propose *sparse metric* technique and a branch-and-cut algorithm to find the optimal β-vertex disruptor in Section 3.3. Section 3.4 presents the simulation results comparing the performance of the proposed approximation algorithms and the exact branch-and-cut algorithm.

2.1 Complexity of Finding Disruptor

In this section we show that both the β-edge disruptor and β-vertex disruptor in directed graph are NP-complete which thus have no polynomial time exact algorithms

unless P = NP. We state a stronger result that both problems are NP-complete even in undirected graph with unit cost edges.

Note that only in this section we consider the problem for undirected graph $G(V, E)$. All results in other sections are studied on directed graphs, thus solving both homogeneous and heterogeneous networks.

2.1.1 NP-completeness of Edge Disruptor

We use a reduction from the balanced cut problem.

Definition 3. *A cut $\langle S, V \setminus S\rangle$ corresponding to a subset $S \in V$ in G is the set of edges with exactly one endpoint in S. The cost of a cut is the sum of its edges' costs (or simply its cardinality in the case all edges have unit costs). We often denote $V \setminus S$ by $\bar{S}$.*

Finding a min cut in the graph is polynomial solvable [71]. However, if one asks for a somewhat "balanced" cut of minimum size, the problem becomes intractable. A balanced cut is defined as following:

Definition 4. *(Balanced cut) An f-balanced cut of a graph $G(V, E)$, where $f : \mathbb{Z}^+ \to \mathbb{R}^+$, asks us to find a cut $\langle S, \bar{S}\rangle$ with the minimum size such that $|S|, |\bar{S}| \geq f(|V|)$.*

Abusing notations, for $0 < c \leq \frac{1}{2}$, we also use c-balanced cut to find the cut $\langle S, \bar{S}\rangle$ with the minimum size such that $\min\{|S|, |\bar{S}|\} \geq c|V|$. We will use the following results on balanced cut shown in [77]:

Corollary 1. *(Monotony) Let g be a function with*

$$0 \leq g(n) - g(n-1) \leq 1$$

Then $f(n) \leq g(n)$ for all n, implies f-balanced cut is polynomially reducible to g-balanced cut.

Corollary 2. *(Upper bound) αn^ϵ-balanced cut is NP-complete for $\alpha, \epsilon > 0$.*

It follows from Corollaries 1 and 2 that for every $f = \Omega(\alpha n^\epsilon)$ f-balanced cut is NP-complete. We are ready to prove the NP-completeness of β-edge disruptor:

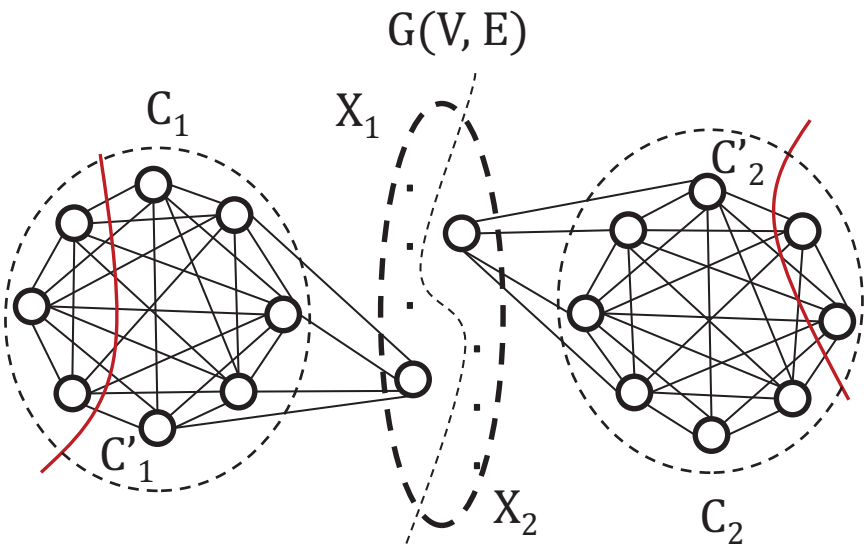


Figure 2-2. Construction of $H(V_H, E_H)$ from $G(V, E)$

Theorem 2.1. *(β-edge disruptor NP-completeness) β-edge disruptor in undirected graph is NP-complete even if all edges have unit weights.*

Proof. We prove the result for the special case when $\beta = \frac{1}{2}$. For other values of β the proof can go through with a slight modification of the reduction. We shall assume that n, the number of nodes is a sufficient large number (for our proof $n > 10^3$).

Consider the decision version of the problem that asks whether an undirected graph $G(V, E)$ contains a $\frac{1}{2}$-edge disruptor of a specified size:

$$\frac{1}{2}\text{-ED} = \{\langle G, K\rangle \mid G \text{ has a } \frac{1}{2}\text{-edge disruptor of size } K\}$$

To show that $\frac{1}{2}$-ED is in NP-complete we must show that it is in NP and that all NP-problems are polynomial time reducible to it. The first part is easy; given a candidate subset of edges, we can easily check in polynomial time if it is a β-edge disruptor of size K. To prove the second part, we show that f-balanced cut is polynomial time reducible to $\frac{1}{2}$-ED where $f = \lfloor \frac{n-\sqrt{2\lfloor \frac{n^2}{3} \rfloor + n}}{2} \rfloor$.

Let $G(V, E)$ be a graph in which one seeks to find a f-balanced cut of size k. Construct the following graph $H(V_H, E_H)$: $V_H = V \cup C_1 \cup C_2$ where C_1, C_2 are two cliques

of size $\lfloor \frac{n^2}{3} \rfloor$. Denote by $N = |V_H| = 2\lfloor \frac{n^2}{3} \rfloor + n$ the total number of nodes in H. In addition to edges in G, C_1, and C_2, connect each vertex $v \in V$ to $\lfloor \frac{n^2}{4} \rfloor + 1$ vertices in C_1 and $\lfloor \frac{n^2}{4} \rfloor + 1$ vertices in C_2 so that degree difference of nodes in the cliques are at most one. We illustrate the construction of $H(V_H, E_H)$ in Figure 2-2.

We show that there is a f-balanced cut of size k in G iff H has an $\frac{1}{2}$-edge disruptor of size $K = n\left(\lfloor \frac{n^2}{4} \rfloor + 1\right) + k$ where $0 \leq k \leq \lfloor \frac{n^2}{4} \rfloor$. Note that the cost of any cut $\langle S, V \setminus S \rangle$ in G is at most $|S||V \setminus S| \leq \lfloor \frac{(|S|+|V\setminus S|)^2}{4} \rfloor = \lfloor \frac{n^2}{4} \rfloor$.

On one hand, an f-balanced cut $\langle S, \bar{S} \rangle$ of size k in G induces a cut $\langle C_1 \cup S, C_2 \cup \bar{S} \rangle$ with size exactly $n\left(\lfloor \frac{n^2}{4} \rfloor + 1\right) + k$. If we select the cut as the disruptor, the pairwise connectivity will be at most $\frac{1}{2}\binom{N}{2}$.

On the other hand, assume that H has an $\frac{1}{2}$-edge disruptor of size $K = n\left(\lfloor \frac{n^2}{4} \rfloor + 1\right) + k$. Remove the edges in the disruptor to reduce the pairwise connectivity to at most $\frac{1}{2}\binom{N}{2}$. Since cutting n nodes in C_1 or C_2 from the cliques requires removing at least $n(\lfloor \frac{n^2}{3} \rfloor - n) > n\left(\lfloor \frac{n^2}{4} \rfloor + 1\right) + k$ edges, let $C_1' \subseteq C_1$ and $C_2' \subseteq C_2$ be giant connected subsets that induce connected subgraphs in C_1 and C_2. These subsets must satisfy $|C_1'| + |C_2'| > |C_1| + |C_2| - n$. Denote by X_1, X_2 the subsets of nodes in V that are connected to C_1' and C_2' respectively. We have $X_1 \cap X_2 = \emptyset$ otherwise C_1' and C_2' will be connected; then, the pairwise connectivity will exceed $\frac{1}{2}\binom{N}{2}$.

We will modify the disruptor without increasing its size and the pairwise connectivity such that no nodes in the the cliques are cut off i.e. we alter the disruptor until $C_1' = C_1$ and $C_2' = C_2$. For each $u \in C_1 \setminus C_1'$ remove from the disruptor all edges connecting u to C_1' and add to the disruptor all edges connecting u to X_2. This will attach u to C_1' while reducing the size of the disruptor at least $(\lfloor \frac{n^2}{3} \rfloor - n) - n$. At the same time select an arbitrary node $v \in X_1$ and add to the disruptor all remaining v's adjacent edges. This increases the size of the disruptor at most $(\lfloor \frac{n^2}{4} \rfloor + 1) + n$ while making v isolated. By doing so we decrease the size of the disruptor by $(\lfloor \frac{n^2}{3} \rfloor - n) - n - ((\lfloor \frac{n^2}{4} \rfloor + 1) + n) > 0$.

In addition, the pairwise connectivity will not increase as we connect u to C'_1 and at the same time disconnect v from C'_1.

If $X_1 = \emptyset$, we can select $v \in X_2$ as in that case $|C'_2 \cup X_2| > |C'_1 \cup X_1|$ that makes sure the pairwise connectivity will not increase. We repeat the same process for every node in $C_2 \setminus C'_2$. Since $|(C_1 \setminus C'_1) \cup (C_2 \setminus C'_2)| < n$, the whole process finishes in less than n steps and results in $C'_1 = C_1$ and $C'_2 = C_2$.

We will prove that $X_1 \cup X_2 = V$ i.e. $\langle X_1, X_2 \rangle$ induces a cut in G. Assume not, the cost to separate $C_1 \cup X_1$ from $C_2 \cup X_2$ will be at least $(\lfloor \frac{n^2}{4} \rfloor + 1)(|V - X_1| + |V - X_2|) = (\lfloor \frac{n^2}{4} \rfloor + 1)(2n - |X_1| - |X_2|) \geq (\lfloor \frac{n^2}{4} \rfloor + 1)(n + 1) > n \left(\lfloor \frac{n^2}{4} \rfloor + 1 \right) + k$ that is a contradiction.

Since $X_1 \cup X_2 = V$ we have that the disruptor induces a cut in G. To have the pairwise connectivity at most $\frac{1}{2}\binom{N}{2}$ both $(C_1 \cup X_1)$ and $(C_2 \cup X_2)$ must have size at least $\frac{N - \sqrt{N}}{2}$. If follows that X_1 and X_2 must have size at least $f(n) = \lfloor \frac{n - \sqrt{2\lfloor \frac{n^2}{3} \rfloor + n}}{2} \rfloor$. The cost of the cut induced by $\langle X_1, X_2 \rangle$ in G will be $n \left(\lfloor \frac{n^2}{4} \rfloor + 1 \right) + k - n(\lfloor \frac{n^2}{4} \rfloor + 1) = k$. □

2.1.2 Hardness of Approximation: Vertex Disruptor

Theorem 2.2. *β-vertex disruptor in undirected graph is NP-complete.*

Proof. We present a polynomial-time reduction from Vertex Cover (VC), an NP-hard problem [40]:

Instance: Given a graph G and a positive integer k.

Question: Does G have a VC of size at most k?

to a decision version of β-vertex disruptor when $\beta = 0$

Instance: Given a graph G and a positive integer k

Question: Does G have a β-vertex disruptor of size at most k when $\beta = 0$?

Pairwise connectivity equals zeros if and only if the complement set of the disruptor is an independent set or in other words the disruptor must be a VC. □

Theorem 2.3. *Unless P = NP, β-vertex disruptor cannot be approximated within a factor of 1.36.*

Proof. We use the same reduction in Theorem 2.2. Assume that we can approximate β-vertex disruptor within a factor less than 1.36 when $\beta = 0$. In [34], Dinur and Safra showed that approximating VC within constant factor less than 1.36 is NP-hard. Since we have an one-to-one mapping between the set of vertex disruptors when $\beta = 0$ and the set of VCs, it follows that we can approximate VC within a factor less than 1.36 (contradiction). □

2.2 Bicriteria Approximation Algorithm for β-edge Disruptor

In this section, we present an $O(\log^{\frac{3}{2}} n)$ pseudo-approximation algorithm for the β-edge disruptor problem in the case when all edges have uniform cost i.e. $c(u, v) = 1 \; \forall (u, v) \in E(G)$. Formally, our algorithm finds in a uniform directed graph G a β'-edge disruptor whose the cost is at most $O(\log^{\frac{3}{2}} n)\text{OPT}_{\beta-ED}$, where $\frac{\beta'}{4} < \beta < \beta'$ and $\text{OPT}_{\beta-ED}$ is the cost of an optimal β-edge disruptor.

As shown in Algorithm 1, the proposal algorithm consists of two main steps. First, we constructs a decomposition tree of G by recursively partitioning the graph into two halves with directed c-balanced cut. Second, we solve the problem on the obtained tree using a dynamic programming algorithm and transfer this solution to the original graph. These two main steps are explained in the next two sections.

2.2.1 Balanced Tree-Decomposition

A tree decomposition of a graph is a recursive partitioning of the node set into smaller and smaller pieces until each piece contains only one single node. We show the tree construction in Algorithm 1 (line 1 to 11). Our decomposition tree is a rooted binary tree whose leaves represent nodes in G. (Because our decomposition tree is a binary tree with n leaves, it will contain exactly $n - 1$ non-leaf nodes. One can prove this with induction on number of nodes.)

Definition 5. *Given a directed graph $G(V, E)$ and a subset of vertices $S \subset V$. We denote the set of edges outgoing from S by $\delta^+(S)$; the set of edges incoming to S by $\delta^-(S)$. A cut $(S, V \setminus S)$ in G is defined as $\delta^+(S)$. A c-balanced cut is a cut $(S, V \setminus S)$ s.t.*

$\min\{|S|, |V \setminus S|\} \geq c|V|$. *The directed c-balanced cut problem is to find the minimum c-balanced cut.*

Note that a cut $(S, V \setminus S)$ separate pairs $(u, v) \in S \times (V \setminus S)$ as paths from v to u cannot exist i.e. no SCC can contain vertex in both S and $V \setminus S$.

The decomposition procedure is as follows. We start with the tree T containing only one root node t_0. We associate the root node t_0 with the vertex set V of G i.e. $V(t_0) = V(G)$. For each node $t_i \in T$ whose $V(t_i)$ contains more than one vertex and $V(t_i)$ has not been partitioned, we partition the subgraph $G_{[V(t_i)]}$ induced by $V(t_i)$ in G using a c-balanced cut algorithm. In detail, we use the directed c-balanced cut algorithm presented in [2] that finds in polynomial time a c'-balanced cut within a factor of $O(\sqrt{\log n})$ from the optimal c-balanced cut for $c' = \alpha c$ and fixed constant α. The constant c is chosen to be $1-\sqrt{\frac{\beta}{\beta'}}$. Create two child nodes t_{i1}, t_{i2} of t_i in T corresponding to two sets of vertices of $G_{[V(t_i)]}$ separated by the cut. We associate with t_i a cut cost $cost(t_i)$ equal to the cost of the c-balanced cut.

We define the root node t_0 to be on level 1. If a node is on level l, all its children are defined to be on level $l + 1$. Note that collections of subsets of vertices in G that correspond to nodes in a same level of T induces a partition in G.

One important parameter of the decomposition tree is the height i.e. the maximum level of nodes in T. Using balanced cuts guarantees a small height of the tree that in turn leads to a small approximation ratio. When separating $V(t_i)$ using the balanced cut, the size of the larger part is at most $(1 - c')|V(t_i)|$. Hence, we can prove by induction that if a node t_i is on level k, the size of the corresponding collection $V(t_i)$ is at most $|V| \times (1 - c')^{k-1}$. It follows that the tree's height is at most $O(-\log_{1-c'} n) = O(\log n)$.

2.2.2 Dynamic Programming Algorithm on the Decomposition Tree

In this section, we present the second main step which uses the dynamic programming to search for the right set of nodes in T that induces an cost-efficient partition in G

Algorithm 1. β**-edge Disruptor**

Input: Uniform edges' weight directed graph $G = (V, E)$

and $0 \leq \beta < \beta' < 1$

Output: A β'-edge disruptor of G.

/* Construct the decomposition tree */

1. $c \leftarrow 1 - \sqrt{\frac{\beta}{\beta'}}$.
2. $T(V_T, E_T) \leftarrow (\{t_0\}, \phi)$, $V(t_0) \leftarrow V(G),\ l(t_0) = 1$
3. **while** $\exists$ unvisited t_i with $|V(t_i)| \geq 2$ **do**
4. Mark t_i visited, create new child nodes t_{i1}, t_{i2} of t_i.
5. $V_T \leftarrow V_T \cup \{t_{i1}, t_{i2}\}$
6. $E_T \leftarrow E_T \cup \{(t_i, t_{i1}), (t_i, t_{i2})\}$
7. Separate $G_{[V(t_i)]}$ using directed c-balanced cut.
8. Associate $V(t_{i1}), V(t_{i2})$ with two separated components.
9. $cost(t_i) \leftarrow$ The cost of the balanced cut

/* Find the minimum cost G-partitionable */

10. Traverse T in post-order, **for** each $t_i \in T$ **do**
11. **for** $p \leftarrow 0$ **to** $\beta' \binom{n}{2}$
12. **if** $\mathcal{P}(G_{[V(t_i)]}) \leq p$ **then** $cost(t_i, p) \leftarrow 0$
13. **else** $cost(t_i, p) \leftarrow \min\{cost(t_{i1}, p_1) + cost(t_{i2}, p_2) + cost(t_i) \mid p_1 + p_2 = p\}$
14. Find $F^{\text{opt}}_{\beta'}$ associating with $T^{\text{opt}}_{\beta'} = \min_{p \leq \beta' \binom{n}{2}} \{cost(t_0, p)\}$
15. Return union of c-balanced cuts at $t_i \in \mathcal{A}(F^{\text{opt}}_{\beta'})$.

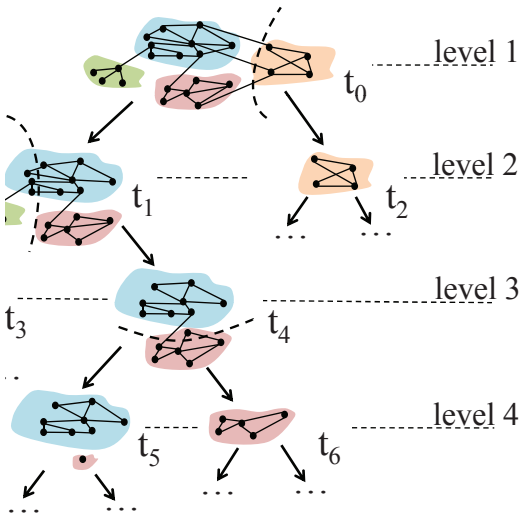


Figure 2-3. A part of a decomposition tree. $F = \{t_2, t_3, t_5, t_6\}$ is a G-partitionable. The corresponding partition $\{V(t_2), V(t_3), V(t_5), V(t_6)\}$ in G can be obtained by using cuts at ancestors of nodes in F i.e. t_0, t_1, t_4.

whose pairwise connectivity is at most $\beta'\binom{n}{2}$. The details of this step are shown in Algorithm 1 (lines 12 to 18).

Denote a set $F = \{t_{u_1}, t_{u_2}, \ldots, t_{u_k}\} \subset V_T$ where V_T is the set of vertices in T so that $V(t_{u_1}), V(t_{u_2}), \ldots, V(t_{u_k})$ is a partition of $V(G)$ i.e. $V(G) = \biguplus_{h=1}^{k} V_{u_h}$. We say such a subset F is G-partitionable. Denote by $\mathcal{A}(t_i)$ the set of ancestors of t_i in T and $\mathcal{A}(F) = \bigcup_{t_i \in F} \mathcal{A}(t_i)$. It is clear that a F is G-partitionable if and only if F satisfies:

1. $\forall t_i, t_j \in F : t_i \notin \mathcal{A}(t_j)$ and $t_j \notin \mathcal{A}(t_i)$
2. $\forall t_i \in V_T, t_i$ is a leaf: $\mathcal{A}(t_i) \cap F \neq \phi$

In case F is G-partitionable, we can separate $V(t_{u_1}), V(t_{u_2}), \ldots, V(t_{u_k})$ in G by performing the cuts corresponding to ancestors of node in F during the tree construction. For example in Figure 2-3, we show a decomposition tree with a G-partitionable set $\{t_2, t_3, t_5, t_6\}$. The corresponding partition $\{V(t_2), V(t_3), V(t_5), V(t_6)\}$ in G can be obtained by cutting $V(t_0), V(t_1), V(t_4)$ successively using balanced cuts in the tree construction. The cut cost, hence, will be $cost(t_0) + cost(t_1) + cost(t_4)$. In general,

the total cost of all the cuts to separate $V(t_{u_1}), V(t_{u_2}), \ldots, V(t_{u_k})$ will be:

$$cost(F) = \sum_{t_u \in \mathcal{A}(F)} cost(t_u)$$

The pairwise connectivity in G then will be:

$$\mathcal{P}(F) = \sum_{t_u \in F} \mathcal{P}(G_{[V(t_u)]})$$

We wish to find F so that $\mathcal{P}(F) \leq \beta' \binom{n}{2}$ i.e. the union of cuts to separate $V(t_{u_1}), V(t_{u_2}), \ldots, V(t_{u_k})$ forms a β'-edge disruptor in G. Because of the suboptimal structure in T, finding such a G-partitionable subset F in V_T with minimum $cost(F)$ can be done in $O(n^3)$ using dynamic programming.

Denote $cost(t_i, p)$ the minimum cut cost to make the pairwise connectivity in $G_{[V(t_i)]}$ equal to p using only cuts corresponding to nodes in the subtree rooted at t_i. The minimum cost for a G-partitionable subset F that induces a β'-edge disruptor of G is then

$$T^{\text{opt}}_{\beta'} = \min_{p \leq \beta' \binom{n}{2}} \{cost(t_0, p)\}$$

where t_0 is the root node in T.

We can easily derive the recursive formula:

$$cost(t_i, p) = \begin{cases} 0 & \text{if } \mathcal{P}(G_{[V(t_i)]}) \leq p \\ \min_{\pi \leq p} cost(t_{i1}, \pi) + cost(t_{i2}, p - \pi) + cost(t_i) & \text{if not} \end{cases}$$

where t_{i1}, t_{i2} are children of t_i.

In the first case, when $\mathcal{P}(G_{[V(t_i)]}) \leq p$ we cut no edges in $G_{[V(t_i)]}$ hence, $cost(t_i, p) = 0$. Otherwise, we try all possible combinations of pairwise connectivity π in $V(t_{i1})$ and $p - \pi$ in $V(t_{i2})$. The combination with the smallest cut cost is then selected.

We now prove that $T^{\text{opt}}_{\beta'} \leq O(\log^{\frac{3}{2}} n)\text{Opt}_{\beta\text{-ED}}$, where $\text{Opt}_{\beta\text{-ED}}$ denotes the cost of the optimal β-edge disruptor in G.

Lemma 1. *There exists a G-partitionable subset of T that induces a β'-edge disruptor whose cost is at most $O\left(\log^{\frac{3}{2}} n\right)$ Opt$_{\beta}$-ED.*

Proof. Let D_β be an optimal β-edge disruptor in G of size Opt$_{\beta}$-ED and $\mathcal{C}_\beta = \{C_1, C_2, ..., C_k\}$ be the set of SCCs, after removing D_β from G.

We construct a G-partitionable subset X_T as in the Algorithm 2. We traverse tree T in preorder i.e. every parent will be visited before its children. For each node t_i, we select t_i into X_T if there exists some component $C_j \in \mathcal{C}_\beta$ that $|V(t_i) \cap C_j| \geq (1-c)|V(t_i)|$ and no ancestors of t_i have been selected into X_T.

We can verify that X_T satisfies two mentioned conditions of a G-partitionable subset.

For each $C_j \in \mathcal{C}_\beta$, define

$$N(C_j) = \{t_i \in T : |V(t_i) \cap C_j| \geq (1-c)|V(t_i)|\}.$$

Since $V(t_i), t_i \in T$ are disjoint subsets. We have

$$\begin{aligned}
\mathcal{P}(X_T) &\leq \sum_{t_i \in X_T} \binom{|V(t_i)|}{2} \\
&= \frac{1}{2} \sum_{C_j \in \mathcal{C}_\beta} \sum_{t_i \in N(C_j)} |V(t_i)|^2 - \frac{n}{2} \\
&\leq \frac{1}{2} \sum_{C_j \in \mathcal{C}_\beta} \left(\sum_{t_i \in N(C_j)} |V(t_i)| \right)^2 - \frac{n}{2} \\
&\leq \frac{1}{2} \sum_{C_j \in \mathcal{C}_\beta} \left(\sqrt{\beta'/\beta} |C_j| \right)^2 - \frac{n}{2} \\
&< \frac{\beta'}{\beta} \frac{1}{2} \left(\sum_{C_j \in \mathcal{C}_\beta} |C_j|^2 - n \right) \leq \beta' \binom{n}{2}
\end{aligned}$$

Finally we show that $cost(X_T) \leq O(\log^{\frac{3}{2}} n)Opt_{\beta}$-ED. Let denote by $h(T)$ the height of T and L_T^i the set of nodes at the ith level in T_G. We have:

$$cost(X_T) = \sum_{i=1}^{h(T)} \sum_{t_u \in \left(L_T^i \cap \mathcal{A}(X_T)\right)} cost(t_u) \tag{2–1}$$

If $t_u \in \mathcal{A}(X_T)$ then t_u is not selected to X_T. Hence, there exists $C_j \in \mathcal{C}$ so that $|V(t_u) \cap C_j| < (1-c)|V(t_u)|$ (otherwise t_u was selected into X_T as it satisfied the conditions in the line 3, Algorithm 2). To guarantee $c < 1 - c$, we need $c < 1/2$ i.e. $\beta > \frac{\beta'}{4}$.

Since the edges in D_β separate C_j from the other SCCs, they also separates $C_j \cap V(t_u)$ from $V(t_u) \backslash C_j$ in $G_{[V(t_u)]}$. Denote by $\delta(t_u, D_\beta)$ the set of edges in D_β separating $C_j \cap V(t_u)$ from $V(t_u) \setminus C_j$ in $G_{[V(t_u)]}$. Obviously, $\delta(t_u, D_\beta)$ is a directed c-balanced cut of $G_{[V(t_u)]}$. Since, the cut we used in the tree construction is only $O(\sqrt{\log n})$ times the optimal c-balanced cut. We have $cost(t_u) \leq O(\sqrt{\log n})|\delta(t_u, D_\beta)|$.

Recall that if two nodes t_u, t_v are on a same level then $V(t_u)$ and $V(t_v)$ are disjoint subsets. It follows that $\delta(t_u, D_\beta)$ and $\delta(t_v, D_\beta)$ are also disjoint sets. Therefore, the cut cost at the ith level

$$\begin{aligned}
& \sum_{t_u \in \left(L_T^i \cap \mathcal{A}(X_T)\right)} cost(t_u) \\
\leq\ & O(\sqrt{\log n}) \sum_{t_u \in \left(L_T^i \cap \mathcal{A}(X_T)\right)} |\delta(t_u, D_\beta)| \\
\leq\ & O(\sqrt{\log n}) \Big| \bigcup_{t_u \in \left(L_T^i \cap \mathcal{A}(X_T)\right)} \delta(t_u, D_\beta) \Big| \\
=\ & O(\sqrt{\log n}) \mathsf{Opt}_{\beta\text{-ED}}
\end{aligned}$$

Since the number of levels $h(T) = O(\log n)$, by Eq. 2–1 we have $cost(X_T) \leq O(\log^{\frac{3}{2}} n)\mathsf{Opt}_{\beta\text{-ED}}$. □

Since there exists a G-partitionable subset of T that induces a β'-edge disruptor whose cost is no more than $O\left(\log^{\frac{3}{2}} n\right)$ $\mathsf{Opt}_{\beta\text{-ED}}$ as shown in Lemma 1 and the dynamic programming always finds the best latent solution in T, the following theorem follows.

Theorem 2.4. *Algorithm 1 achieves a pseudo-approximation ratio of $O(\log^{\frac{3}{2}} n)$ for the β-edges disruptor problem.*

Time complexity: Construction of the decomposition tree takes $O(n^{9.5})$. The major portion of time is for solving an semidefinite programming with $\Omega(n^3)$ constraints. Finding

Algorithm 2. *Find a good G-partitionable subset of T that induces a β'-edge disruptor in G*

Initialization: $X_T \leftarrow \phi$; `Preorder-Selection(`t_0`).`

`Preorder-Selection(`t_u`)`

1: **if** $(\exists C_j \in \mathcal{C}_\beta : |V(t_u) \cap C_j| \geq (1-c)|V(t_u)|)$ **then**
2: $\quad X_T \leftarrow X_T \cup \{t_u\}$
3: **else** let t_{u1}, t_{u2} be children of t_u,
4: $\quad$ `Preorder-Selection(`t_{u1}`)`
5: $\quad$ `Preorder-Selection(`t_{u2}`)`
6: **end if**

the optimal solution using Dynamic Programming takes $O(n^3)$. Hence, the overall time complexity is $O(n^{9.5})$.

2.3 Bounds on the Size of Edge Disruptor

Simultaneous attacks can cause devastating damage, breaking down communication networks into small fragments. To mitigate the risk and develop proactive responses, it is essential to assess the robustness of network in the worst-case scenarios. In this paper, we propose a spectral lower-bound on the number of removed links to incur a certain level of disruption in terms of *pairwise connectivity*. Our lower-bound explores the latent structural information in the *network Laplacian spectrum*, the set of eigenvalues of the Laplacian matrix, to provide guarantees on the robustness of the network against intentional attacks. Such guarantees often cannot be found in heuristic methods for identifying critical infrastructures. For the first time, the attack-resistant proofs of large scale communication networks against link attacks are presented.

Connectivity plays a vital role in network performance and is fundamental to vulnerability assessment. The number of connected node pairs in the network, (a.k.a pairwise connectivity), lends itself as an effective measure to account for the effect of the attacks [11, 14, 15, 30, 33, 62, 64].

Vulnerability assessment has been recently formulated as an connectivity optimization problem called β-*edge disruptor*, which finds *a minimum cost links* whose removal causes a *significant level (β) of network pairwise degradation* [33]. The β-edge disruptor reflects the common sense that when breaking the network by removing links, the more links required to be removed, the less vulnerable the network is. The β-edge disruptor approach enables the exploration of different network disruption levels which can be used to gain the deeper insight into network structure and robustness in various operating environments.

Unfortunately, the β-edge disruptor problem is NP-hard [33] i.e. there is no efficient algorithm to solve the problem, unless P=NP. A pseudo-approximation algorithm and mathematical approaches for the β-edge disruptor problems are introduced in [33] and [30], respectively. Although those methods can provide performance guarantees, they are only applicable for small and medium networks of few thousand nodes. For larger networks, we have to rely on heuristics which can have arbitrary bad worst-case performance. Hence, there is a lack of methods to provide robustness proofs against intentional attacks for large networks.

In this paper, we analyze the network spectrum, the eigenvalues of the Laplacian matrix, to give a *lower-bound* for the minimum size of a β-edge disruptor, thus, give a certificate on the robustness of the network. Our spectral bound is formulated as an optimization problem of the Laplacian eigenvalues, which are known to contain rich information about the topological structure [23].

Since exact measurement for the β-edge disruptor is not available in general, our lower-bound can be coupled with upper bound methods[1] to narrow down the range for actual vulnerability/robustness of the network. We emphasize that while upper bounds for β-edge disruptor (or any other minimization problem) can be designed easily,

[1] Each heuristic to find β-edge disruptor is an upper bound for the problem

techniques for deriving lower-bound is much scattered in literature. Our contributions are summarized as follows.

- We introduce a new spectral lower-bound for the β-edge disruptor problem in form of an eigenvalue optimization problems. At the same time, we enrich the literature on lower-bound techniques.
- We present two efficient methods to compute the proposed lower-bound: 1) the Lagrange multiplier method and 2) the dynamic programming algorithm. Moreover, the Lagrange multiplier method can derive the lower-bound with only a small number of smallest eigenvalues. This is important for large networks where computing the whole network spectrum is both time and memory consuming.
- We perform experiments on different network types and real large-scale networks to demonstrate the quality of the proposed lower-bound and quantify the robustness of the studied networks against intentional attacks.

Organization. We briefly present terminologies and problem definitions in subsection 4.1. In subsection 2.3.2, we introduce the spectral lower-bound for the the β-disruptor problem together with two methods to compute the lower-bound. Experimental results on different network models and real network instances are obtained in subsection 6.4. Finally, we conclude the paper in subsection **??**.

2.3.1 Laplacian Matrix and and Its Eigenvalues

We abstract our general network model as a graph $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ refers to a set of nodes and E refers to a set of links. Each edge $(v_i, v_j) \in E$ has a removal cost $c_{ij} \geq 0$ (and $c_{ij} = 0$ if $(v_i, v_j) \notin E$). For convenience, we also denote the number of nodes and links by n and m, respectively.

Since the main purpose of network lies in connecting all the interacting elements in the network, we study on the *overall pairwise connectivity*, which is defined as the number of connected vertex pairs in G. If G is an undirected graph, a vertex pair $(u, v) \in V \times V$ is connected iff there exists a path between u and v. We denote the pairwise connectivity of a graph G by $\mathcal{P}(G)$. Apparently, the pairwise connectivity is maximized at $\binom{n}{2}$ when G is a (strongly) connected graph.

Let $\boldsymbol{A} = \{c_{ij}\}$ be the weighted *adjacency matrix* and $\boldsymbol{D}$ be the *degree matrix*, defined as the diagonal matrix with the weighted degrees $d_1, d_2, \ldots, d_n$ on the diagonal, where $d_i = \sum_j c_{ij}$.

The unnormalized graph Laplacian matrix [61] is defined as

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$$

The matrix L is symmetric and positive semi-definite, since for every vector $x \in \mathbb{R}^n$ we can verify that

$$x^T \boldsymbol{L} x = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2 \geq 0. \tag{2–2}$$

A direct consequence is that $\boldsymbol{L}$ has n non-negative, real-valued eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$. In addition, the smallest eigenvalue of λ_1 is zero and the corresponding eigenvector is the constant one vector $\mathbb{1}$ [61].

The second smallest eigenvector λ_2 is known as the algebraic connectivity of the graph and can be used to describe many properties of graphs [61]. For example, the graph G is connected if and only if $\lambda_2 > 0$. For β-edge disruptor problem, the following lower-bound can be derived from λ_2.

Lemma 2. *For any connected graph G, we have*

$$\mathrm{OPT}_\beta \geq \frac{1-\beta}{2} \lambda_2 (n-1) \tag{2–3}$$

where OPT_β denotes the minimum size of a β-edge disruptor.

However, the bound provided in Eq. 2–3 is rather loose, as the value of λ_2 is often very close to zero (for example when bridges, edges whose deletion increases the number of connected components, are presented in the networks.) This motivates us to study higher eigenvalues beyond λ_2 to design stronger bound for the β-edge disruptor problem.

2.3.2 Spectral Lower-bound for Link Assessment

In this subsection, we derive a lower-bound on size of β-edge disruptor using higher eigenvalues of the Laplacian matrix L. We first formulate the lower-bound as an eigenvalue optimization problem. Then two methods with different trade-off between time and quality are introduced to compute the lower-bound.

Let E^*_β be an optimal β-edge disruptor and $s^*_1 \geq s^*_2 \geq \ldots \geq s^*_n$ be the sizes of the connected components after removing E^*_β from the network. Then we can relate OPT_β to the size of the components via the following lemma.

Lemma 3. *[35] Let a k-partition of a graph be a division of the vertices into k disjoint subsets containing $s_1 \geq s_2 \geq \ldots \geq s_k$ vertices. Let E_{cut} be the set of edges whose two vertices belong to different subsets. Let $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_k$, be the k smallest eigenvalues of the Laplacian matrix plus any diagonal matrix U such that the sum of all the elements of U is zero. Then*

$$|E_{\text{cut}}| \geq \frac{1}{2} \sum_{i=1}^{k} s_i \lambda_i.$$

Thus, we have $\text{OPT}_\beta = |E^*_\beta| \geq \frac{1}{2} \sum_{i=1}^{n} s^*_i \lambda_i$. Here we allow imaginary subsets of size zero and assume w.l.o.g. that $k = n$. Note that $s^*_1, \ldots, s^*_n$ are not known without finding E^*_β. Thus, we consider all possible values of $\{s_1, \ldots, s_n\}$ which infer network partitions of pairwise connectivity at most $\beta\binom{n}{2}$, and get the minimum of the sum $\frac{1}{2} \sum_{i=1}^{n} s_i \lambda_i$ as a lower-bound on OPT_β.

Formally, our spectral lower-bound on OPT_β is given by solving the following quadratic programming (QP) optimization problem.

$$\text{minimize} \quad \frac{1}{2}\sum_{i=1}^{n} s_i\lambda_i \tag{2–4}$$

$$\text{subject to} \quad \sum_{i=1}^{n} s_i = n \tag{2–5}$$

$$\sum_{i=1}^{n} \binom{s_i}{2} \leq \beta\binom{n}{2} \tag{2–6}$$

$$s_i \in \{0, 1, \ldots, n\} \tag{2–7}$$

Theorem 2.5. *Let Q_β be the optimal objective of the QP problem (2–9-2–12) and OPT_β be the minimum β-edge disruptor of graph $G = (V, E)$. Then, $Q_\beta \leq \mathrm{OPT}_\beta$ for $\beta \in [0, 1]$. Moreover, the equality holds when $\beta = 0$ or $\beta = 1$*

Proof. As discussed in the previous paragraph, the sizes of connected components after removing optimal β-edge disruptor satisfy all constraints (2–5-2–7). Hence, $Q_\beta \leq \mathrm{OPT}_\beta$ for all $\beta \in [0, 1]$. We continue with the tightness of the bound at extreme cases when $\beta = 0$ and $\beta = 1$.

Case $\beta = 0$: all subsets are of size one. Hence, $Q_1 = \frac{1}{2}\sum_{i=1}^{n} \lambda_i = \frac{1}{2}\text{Trace}(\boldsymbol{L}) = \frac{1}{2}(2|E|) = |E|$. The only way to cut all pairs in the network is to cut all edges. In other words, $Q_0 = \mathrm{OPT}_0 = |E|$.

Case $\beta = 1$: in order to achieve the maximum connectivity $\binom{n}{2}$, there must be a single partition in the network and the optimal disruptor cutting no edges. That is $s_1 = n$ and $s_i = 0\ \forall i > 1$. Since $\lambda_1 = 0$, it follows that $Q_1 = 0 = \mathrm{OPT}_1$. □

Since s_i are integral values, we propose a dynamic programming algorithm to compute the spectral bound in next subsubsection.

2.3.2.1 Dynamic Programming Method

We first describe the optimal solution structure for the optimization problem in (2–9-2–12).

Lemma 4. *There exists an optimal solution $\boldsymbol{s}^*$ of* QP*(2–4-2–7) such that $s_1^* \geq s_2^* \geq \ldots \geq s_n^*$.*

Proof. Let $\boldsymbol{s}^* = \{s_1^*, s_2^*, \ldots, s_n^*\}$ be an optimal solution of QP(2–9-2–12). Denote inv$(\boldsymbol{s}^*)$ the number of inversions of m^* i.e. such pairs of indices (i, j) that $i < j$ such that $s_i^* > s_j^*$. If inv$(\boldsymbol{s}^*) = 0$, then $s_1^* \geq s_2^*, \geq \ldots \geq s_n^*$, otherwise there exists a pair $i < j$ and $s_i^* > s_j^*$. Construct $\boldsymbol{s}'$ by swapping s_i^* and s_j^* inside $\boldsymbol{s}^*$. Then, $\boldsymbol{s}'$ is a feasible solution of QP(2–9-2–12) and the objective increases an amount $s_i^*\lambda_j + s_j^*\lambda_i - (s_i^*\lambda_i + s_j^*\lambda_j) = (s_i^* - s_j^*)(\lambda_j - \lambda_i) \geq 0$. Thus, we obtain a new optimal solution with less the number of inversions. Repeat the process at most $\binom{n}{2}$, that is the maximum number of inversions in $\boldsymbol{s}^*$, we finally obtain an optimal solution with no inversions. That optimal solution shall satisfy the lemma's condition. □

Algorithm 3: ILB(G, β)

Compute $\lambda_1, \ldots, \lambda_n$

$\mathcal{L}_k(l,p) = \begin{cases} +\infty, & \text{if } p < p_{\min}(l,k) \\ \lambda_1 l = 0, & \text{if } p \geq p_{\max}(l,k) \end{cases}$

for $k = 1$ **to** n

  for $l = 1$ **to** n

    for $p = p_{\min}(l,k)$ **to** $\min\left\{\beta\binom{n}{2}, p_{\max}(l,k)\right\}$

      $\mathcal{L}_k(l,p) = \min\left\{\begin{array}{l} \mathcal{L}_{k-1}(l,p), \\ \mathcal{L}_k(l-k, p-l+k) + \sum_{i=1}^k \lambda_i \end{array}\right\}$

  if $\mathcal{L}_{k-1}\left(n, \beta\binom{n}{2}\right) = \mathcal{L}_k\left(n, \beta\binom{n}{2}\right)$

    return $\lceil \mathcal{L}_k\left(n, \beta\binom{n}{2}\right)\rceil$

return $\lceil \mathcal{L}_n\left(n, \beta\binom{n}{2}\right)\rceil$

For $k \leq l \leq n$ and $p \leq \beta\binom{n}{2}$, define $\mathcal{L}_k(l,p)$ to be the minimum spectral bound obtained by first k subsets that the total sizes is l and the total pairwise connectivity is at most p. That is

$$\mathcal{L}_k(l,p) = \min_{\boldsymbol{s}^{(k)} \in \mathbb{N}^k} \left\{ \boldsymbol{s}^{(k)T} \boldsymbol{\lambda}^{(k)} \;:\; \|\boldsymbol{s}^{(k)}\|_1 = l,\ \sum_{i=1}^k \binom{s_i}{2} \leq p \right\},$$

Then the optimal objective value QP(2–4-2–7) shall be given by $Q_\beta = \mathcal{L}_n(n, \beta\binom{n}{2})$.

By Lemma 13, we pay attention only to partitions satisfying $s_1 \geq s_2 \geq \ldots \geq s_n$. We now derive the recursive formula for $\mathcal{L}_p(l,k)$ based on the sub-optimal structure of the QP problem. Consider two possible cases of s_k

- $s_k = 0$: There are at most $k-1$ partitions whose sizes sum up to l. Hence, for this case $\mathcal{L}(l,k) = \mathcal{L}_{k-1}(l,p)$.
- $s_k > 0$: Since $s_1 \geq s_2 \geq \ldots \geq s_k > 0$. Let $\tilde{s}_i = s_i - 1 \geq 0$, the vector $\tilde{s} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_k\}$ satisfies simultaneously the following

$$\begin{aligned}
\sum_{i=1}^{k} \lambda_i \tilde{s}_i &= \sum_{i=1}^{k} \lambda_i s_i - \sum_{i=1}^{k} \lambda_i \\
\sum_{i=1}^{k} \tilde{s}_i &= \sum_{i=1}^{k} s_i - k = l - k \\
\sum_{i=1}^{k} \binom{\tilde{s}_i}{2} &= \sum_{i=1}^{k} \left[\binom{s_i}{2} - s_i + 1 \right] \\
&= \sum_{i=1}^{k} \binom{s_i}{2} - l + k \leq p - l + k
\end{aligned}$$

  Therefore, in this case $\mathcal{L}_k(l,p) = \mathcal{L}_k(l-k, p-l+k) + \sum_{i=1}^{k} \lambda_i$
  In summary, we have

$$\mathcal{L}_k(l,p) = \min \left\{ \begin{array}{l} \mathcal{L}_{k-1}(l,p), \\ \mathcal{L}_k(l-k, p-l+k) + \sum_{i=1}^{k} \lambda_i \end{array} \right\}$$

We compute value of $\mathcal{L}_p(l,k)$ in increasing order of p and l but in decreasing order of k. The base cases for $\mathcal{L}_p(l,k)$ are as follow.

$$\mathcal{L}_k(l,p) = \begin{cases} +\infty, & \text{if } p < p_{\min}(l,k) \\ \lambda_1 l = 0, & \text{if } p \geq p_{\max}(l,k) \end{cases} \tag{2–8}$$

where $p_{\min}(l,k) = \binom{\lceil l/k \rceil}{2}(l \bmod k) + \binom{\lfloor l/k \rfloor}{2}(k - l \bmod k)$ and $p_{\max}(l,k) = \binom{l}{2}$ that are the minimum and maximum pairwise connectivity of a graph with l vertices and k connected components, respectively.

Theorem 2.6. *Optimal solutions of* QP*(2–4-2–7) can be found in* $O(n^4)$ *time and* $O(n^3)$ *space.*

Thus, the spectral bound can be computed in polynomial time. However, the high time complexity of the dynamic programming algorithm prevents the method from being applied to large networks. Moreover, the dynamic programming algorithm requires computing the whole set of eigenvalues of the networks, which is both time and memory consuming. We continue with an approximation of the spectral bound that achieves (almost) the same lower-bound quality in significantly less time.

2.3.2.2 Lagrange Multipliers Method

We relax the integral conditions on s_i to obtain the following relaxation of the QP, rewritten in vector notation.

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{s}^T\boldsymbol{\lambda} && \text{(2–9)} \\ \text{subject to} \quad & \|\boldsymbol{s}\|_1 - n = 0, && \text{(2–10)} \\ & \|\boldsymbol{s}\|_2^2 - \Delta_\beta \leq 0, && \text{(2–11)} \\ & \boldsymbol{s} \geq 0, && \text{(2–12)} \end{aligned}$$

where $\Delta_\beta = \beta n(n-1) + n$ and $\|.\|$ denotes the Euclidean norm.

The Lagrange multiplier is then

$$\mathcal{L}(\boldsymbol{s}, \chi, \psi, \boldsymbol{\omega}) = \frac{1}{2}\boldsymbol{s}^T\boldsymbol{\lambda} + \chi(\|\boldsymbol{s}\|_1 - n) + \psi(\|\boldsymbol{s}\|_2^2 - \Delta_\beta) - \boldsymbol{\omega}^T\boldsymbol{s}$$

where $\boldsymbol{\omega} = (\omega_1, \ldots, \omega_n) \geq 0$ is a positive multiplier vector.

Notice that the problem is a convex optimization problem with differentiable objective and constraint functions and it satisfies the Slater's condition with $\boldsymbol{s} = (1, 1, \ldots, 1)^T$ [19]. Hence, the following Karush–Kuhn–Tucker (KKT) conditions provide

the necessary and sufficient conditions for optimality

$$\begin{aligned}
\nabla_{\boldsymbol{s}}\mathcal{L} = \frac{1}{2}\boldsymbol{\lambda} + \chi + 2\psi\boldsymbol{s} - \boldsymbol{\omega} &= 0 \\
\nabla_{\chi}\mathcal{L} = \|\boldsymbol{s}\|_1 - n &= 0 \\
\nabla_{\psi}\mathcal{L} = \|\boldsymbol{s}\|_2^2 - \Delta_\beta &= 0 \\
\boldsymbol{\omega}^T\boldsymbol{s} &= 0 \\
\boldsymbol{s}, \psi, \boldsymbol{\omega} &\geq 0
\end{aligned}$$

Algorithm 4: LMB(G, β)

$t = \lceil 2/\beta \rceil, \quad \Delta_\beta \leftarrow \lfloor \beta n(n-1) + n \rfloor$
Compute $\lambda_1, \ldots, \lambda_t$
for $k = 1$ **to** n
  if $k > t$ **then**
    $t = \min\{2t, n\}$
    Compute $\lambda_1, \ldots, \lambda_t$
  Compute ψ as in Eq. 2–20.
  Compute $\mathcal{D}_\beta^{(k)}$, and $\mathcal{C}_\beta^{(k)}$ as in Eqs. 2–21, and 2–22
  if ($\psi \geq 0$ and $\mathcal{C}_\beta^{(k)} \geq 0$) or ($k = n$) **then**
    return $\lceil \mathcal{D}_\beta^{(k)} \rceil$
end for

Let $k = \max\{i \mid s_i > 0\}$. By Lemma 13 and the complementary slackness $\boldsymbol{\omega}^T\boldsymbol{s} = 0$, we have $s_i > 0$ for $i \leq k$, thus, $s_i = 0 \; \forall i > k$ and $\omega_j = 0 \; \forall j \leq k$.

Denote $\boldsymbol{s}^{(k)} = \{s_1, s_2, \ldots, s_k\}$ and $\boldsymbol{\lambda}^{(k)} = \{\lambda_1, \lambda_2, \ldots, \lambda_k\}$, the KKT condition can be simplified to

$$\nabla_{\boldsymbol{s}^{(k)}} \mathcal{L} = \frac{1}{2}\boldsymbol{\lambda}^{(k)} + \chi + 2\psi \boldsymbol{s}^{(k)} = 0, \qquad i \leq k \tag{2–13}$$

$$\nabla_{s_i} \mathcal{L} = \frac{1}{2}\lambda_i + \chi - \omega_i = 0, \qquad i > k \tag{2–14}$$

$$\nabla_{\chi} \mathcal{L} = \|\boldsymbol{s}^{(k)}\|_1 - n = 0, \tag{2–15}$$

$$\nabla_{\psi} \mathcal{L} = \|\boldsymbol{s}^{(k)}\|_2^2 - \Delta_\beta = 0, \tag{2–16}$$

$$\boldsymbol{s}^{(k)} > 0, \psi > 0, \boldsymbol{\omega}^{(k)} = 0 \tag{2–17}$$

For each value of k, we can solve for values of s_i and check if all $s_i \geq 0$. The other unknowns can be found as follows. First, substitute the constraint (2–15) into the sum of the constraints (2–13) to obtain χ in terms of ψ.

$$\chi = -2\frac{n}{k}\psi - \frac{\|\boldsymbol{\lambda}^{(k)}\|_1}{2k} \tag{2–18}$$

Therefore, we can derive $\boldsymbol{s}^{(k)}$ from (2–13) as

$$\boldsymbol{s}^{(k)} = \frac{n}{k} + \left(\frac{\|\boldsymbol{\lambda}^{(k)}\|_1}{4k} - \frac{\boldsymbol{\lambda}^{(k)}}{4}\right)\frac{1}{\psi} \tag{2–19}$$

Substituting the above equation into the condition (2–16) and solving for ψ, we have

$$\begin{aligned} &\|\boldsymbol{s}^{(k)}\|_2^2 - \Delta_\beta = 0 \\ \Leftrightarrow &\left(\frac{\|\boldsymbol{\lambda}^{(k)}\|_2^2}{16} - \frac{\|\boldsymbol{\lambda}^{(k)}\|_1^2}{16k}\right)\frac{1}{\psi^2} = \Delta_\beta - \frac{n^2}{k} \\ \Leftrightarrow &\psi = \frac{1}{4}\left(\frac{\|\boldsymbol{\lambda}^{(k)}\|_2^2 - \|\boldsymbol{\lambda}^{(k)}\|_1^2/k}{\Delta_\beta - \frac{n^2}{k}}\right)^{1/2} \end{aligned} \tag{2–20}$$

The objective is then

$$\begin{aligned} \mathcal{D}_\beta^{(k)} &= \frac{1}{2}\boldsymbol{s}^{(k)T}\boldsymbol{\lambda}^{(k)} = n\frac{\|\boldsymbol{\lambda}^{(k)}\|_1}{2k} + \left(\frac{\|\boldsymbol{\lambda}^{(k)}\|_1^2}{4k} - \frac{\|\boldsymbol{\lambda}^{(k)}\|_2^2}{4}\right)\frac{1}{2\psi} \\ &= n\frac{\|\boldsymbol{\lambda}^{(k)}\|_1}{2k} - \frac{1}{2}\left(\|\boldsymbol{\lambda}^{(k)}\|_2^2 - \frac{\|\boldsymbol{\lambda}^{(k)}\|_1^2}{k}\right)^{1/2}\left(\Delta_\beta - \frac{n^2}{k}\right)^{1/2} \end{aligned} \tag{2–21}$$

Since $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$, Eq. 2–19 implies that $s_1^{(k)} \geq s_2^{(k)} \geq \ldots \geq s_k^{(k)}$. Hence, in order to satisfy $\boldsymbol{s}^{(k)} > 0$, it is sufficient that

$$\mathcal{C}_\beta^{(k)} = s_k^{(k)} = \frac{n}{k} + \left(\frac{\|\boldsymbol{\lambda}^{(k)}\|_1}{4k} - \frac{\boldsymbol{\lambda}_k}{4} \right) \frac{1}{\psi} \geq 0. \tag{2–22}$$

Theorem 2.7. *The size of a β-edge disruptor is lower-bounded by*

$$\mathcal{D}_\beta = \min_{n \geq k \geq n^2/\Delta_\beta} \left\{ \mathcal{D}_\beta^{(k)} \mid \mathcal{C}_\beta^{(k)} > 0 \right\},$$

where $\mathcal{D}_\beta^{(k)}$ and $\mathcal{C}_\beta^{(k)}$ are given by Eqs. 2–21 and 2–22.

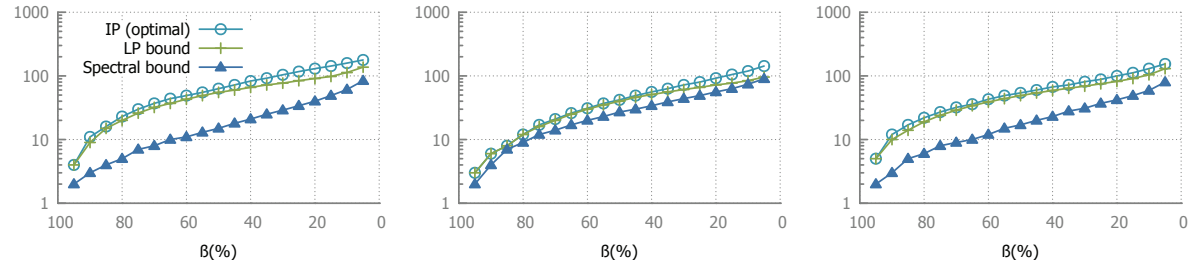
The steps to solve the relaxation of the QP is summarized in the Algorithm 4 (LMB Algorithm).

Time complexity. The LMB algorithm spends its major time on computing the eigenvalues. This can be done with Implicitly Restarted Lanczos Method which has worst-case time complexity $O(mKh+nK^2h+K^3h)$ where K is the number of eigenvalues to be computed, and h is the number of iterations for the eigenvalue algorithm to converge [80]. Given the eigenvalues, the rest of LMB takes only $O(n)$ time in the worst-case.

The number of required eigenvalues K is small in our algorithm. At beginning, the algorithm computes $t = \lceil 2/\beta \rceil$ smallest eigenvalues and the number of computed eigenvalues is double each time if necessary. In our experiments, the number of needed eigenvalues is $2/\beta$ in most cases. For example, to bound the number of necessary links whose removal disrupts 90% pairwise connectivity we only need to compute about 20 smallest eigenvalues of the Laplacian matrix. We found the LMB algorithm to be scalable, taking *linear time* with respect to the number of nodes and edges.

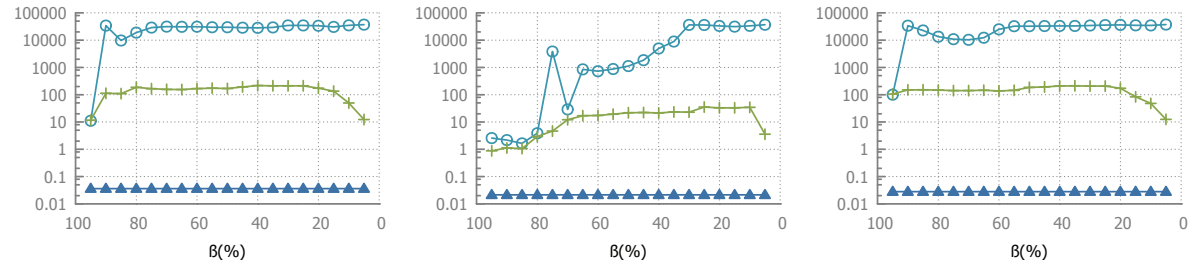
2.3.2.3 Time and quality trade-off

On one hand, the ILB algorithm (Algorithm 3) provides a better bound than that of the LMB algorithm. The reason is that ILB solves for exact solutions of the QP while



A Erdos-Reyni (random) network B Barabasi (power-law) network C Watts-Strogatz network

Figure 2-4. Minimum cost and lower-bounds for β-disruptor on the synthesis networks



A Erdos-Reyni (random) network B Barabasi (power-law) network C Watts-Strogatz network

Figure 2-5. Running time on the synthesis networks

LMB only targets a relaxation of the QP. However, the difference between the output of two algorithms is negligible small and either zero or one [2] in our experiments.

On the other hand, the LMB has much more practical time complexity. The ILB has high time complexity $O(n^4)$ and can only applied for network up to few thousand nodes. In contrast, LMB takes only linear time to compute its competitive bound. Overall for small and medium networks, one can apply ILB algorithm (or other mathematical approaches [30]) to compute the lower-bound, however, for large networks LMB remains the only choice.

2.3.3 Experimental Results

We compute our spectral lower-bound for both synthetic and real-world networks and compare the results with the optimal results whenever possible.

2.3.3.1 Synthetic Networks

We generate the synthetic networks following well-known complex network models. All networks have 100 nodes and around 300 edges. The details of those networks are as follows.

- **Erdos-Reyni**: A random graph of 100 vertices and 300 edges following the Erdos-Reyni model [36].
- **Barabsi-Albert**: A power-law model using preferential attachment mechanism [12].
- **Small world**: A random graph following Watts and Strogatz model [79]. The dimension of the lattice is set to be 3 and the rewiring probability is 0.3.

The optimal solutions are found with the integer programming using the sparse metric technique in [30]. The technique in [30] is also applied to compute the lower-bound given by solving the linear programming. The results produced by ILB and LMB algorithms are identical (after rounded up) and plotted under the same name "spectral bound". All algorithms were run on a PC with Intel Xeon 2.93 Ghz processor and 12 GB memory. The integer programming (IP) and the linear programming (LP) are solved with the mathematical optimization package GUROBI 4.5.

The minimum number of links whose removal causes certain level of disruption, are shown in Fig. 4-4. For all three different networks, solving LP gives good lower-bound on the minimum number of links to remove. The spectral bounds are much worse than the LP bounds in the random and small-world networks; however, the spectral bound

[2] Both algorithms round up their results to the nearest integers.

closely approaches the LP bounds and the optimal solution when the network has the power-law topology of the Barabasi model.

As shown in Fig. 4-5, there is a big gap between the running time of the spectral bound and those of LP and IP. Note that all the spectral bound are computed at once, i.e., the provided running time is the total running time over all different values of β. Even though the running time of the spectral bound is still thousand of times faster than LP and IP.

Overall, while IP is best used for small networks, and LP can be used for medium networks of few thousand nodes, the only feasible method to compute the lower-bound in large networks is the spectral bound. One of the attractive aspect of the LMB spectral bound, described in the Alg. 2, is that the algorithm can be easily implemented in a distributed manner. The most time-consuming part of the algorithm is to compute the few smallest eigenvalues. This can be done distributedly with the existing mathematical software [16].

Table 2-1. Sizes of the investigated networks and the corresponding running time to compute the lower-bound

	CAIDA AS	**Oregon AS**	**P2P Gnutella**
Vertices	8,020	11,174	22,663
Edges	36,406	23,410	109, 386
Time (s)	1530.1	321.0	207.9

2.3.3.2 Real-world Datasets

We compute the spectral lower-bounds for real networks are shown in Fig. 2-6. Neither LP nor IP can run on these networks due to both time and memory limits. The studied networks are

- **Gnutella P2P**: Gnutella peer-to-peer network from from Aug. 25, 2002 [56]. Nodes represents hosts in the network and edges are the connections between the Gnutella hosts.
- **Oregon AS**: AS peering information inferred from Oregon route-views between Mar. 31 and May 26, 2001 [56].

- **CAIDA AS**: The CAIDA AS Relationships Datasets, from September 17, 2007 [56].

The lower-bounds in Fig. 2-6 indicates that it is difficult to destroy major connectivity in communication networks. For examples, even after removing 369 links at least 50% node pairs in the CAIDA AS network stay connected; and to bring down the connectivity level in the Gnutella P2P network to 15% one has to destroy at least 960 links. Due to low edge density, the Oregon AS network tends to be more vulnerable than the other two networks. Nevertheless, uterly disrupting the connectivity in the network to 5% level would require removing more than 763 links.

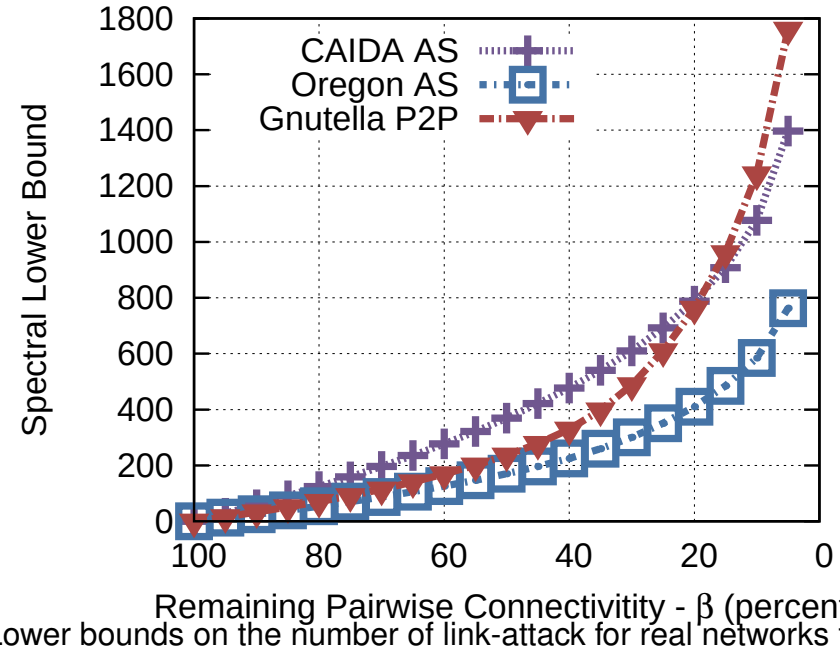


Figure 2-6. Lower bounds on the number of link-attack for real networks found with the LMB algorithm.

CHAPTER 3
MULTIPLE NODE ATTACKS

3.1 Bicriteria Approximation Algorithm for β-vertex Disruptor

We present a polynomial time algorithm (Algorithm 3) that finds a β'-vertex disruptor in the directed graph $G(V, E)$ whose the size is at most $O(\log n \log\log n)$ times the optimal β-vertex disruptor where $0 < \beta < \beta'^2$. The algorithm involves in two phases. In the first phase, we split each vertex $v \in V$ into two vertices v^+ and v^- while putting an edge from v^- to v^+ and show that removing v in G has the same effects as removing edge $(v^+ \rightarrow v^-)$ in the new graph. In the second phase, we try to decompose the new graph into SCCs capping the sizes of the largest component while minimizing the number of removed edges. We relax the constraints on the size of each component until the set of cut edges induces a β'-vertex disruptor in the original graph G.

Given a directed graph $G(V, E)$ for which we want to find a small β'-vertex disruptor, we split each vertex in G into two new vertices to obtain a new directed graph $G'(V', E')$ where

$$\begin{aligned} V' &= \{\, v^-, v^+ \mid v \in V\} \\ E' &= \{(v^- \rightarrow v^+) \mid v \in V\} \\ &\quad \cup \{(u^+ \rightarrow v^-) \mid (u \rightarrow v) \in E\} \end{aligned}$$

The new graph $G'(V', E')$ will have twice the number of vertices in G i.e. $|V'| = 2|V| = 2n$. An example for the first phase is shown in Figure 3-1.

We set the costs of all edges in $E'_V = \{(v^- \rightarrow v^+) \mid v \in V\}$ to 1 and other edges in E' to $+\infty$ so that only edges in E'_V can be selected in an edge disruptor set. In implementation, it is safe to set the costs of edges not in E'_V to $O(n)$ noting that by paying a cost of $2n$ we can effectively disconnect all edges in E'_V.

Consider a directed edge disruptor set $D'_e \subset E'$ that contains only edge in E'_V. We have a one-to-one correspondence between D'_e to a set $D_v = \{v \mid (v^- \rightarrow v^+) \in$

Algorithm 5. β'**-vertex disruptor**

Input:Directed graph $G = (V, E)$ and fixed $0 < \beta' < 1$.

Output:β'-vertex disruptor of G

1. $G'(V', E') \leftarrow (\phi, \phi)$
2. $\forall v \in V : V' \leftarrow V' \cup \{v^+, v^-\}$
3. $\forall v \in V : E' \leftarrow E' \cup \{(v^- \rightarrow v^+)\}, c(v^-, v^+) \leftarrow 1$
4. $\forall (u \rightarrow v) \in E : E' \leftarrow E' \cup \{u^+ \rightarrow v^-\}, c(u^+, v^-) \leftarrow \infty$
5. $\underline{\beta} \leftarrow 0, \overline{\beta} \leftarrow 1$
6. $D_V \leftarrow V(G)$
7. **while** $(\overline{\beta} - \underline{\beta} > \epsilon)$ **do**
8. $\tilde{\beta} \leftarrow \lfloor \frac{\underline{\beta} + \overline{\beta}}{2\epsilon} \rfloor \times \epsilon$
9. Find $D_e \subset E'$ to separate G' into strongly connected components of sizes at most $\tilde{\beta}|V'|$ using algorithm in [37]
10. $D_v \leftarrow \{v \in V(G) \mid (v^+ \rightarrow v^-) \in D_e\}$
11. **if** $\mathcal{P}(G_{[V \setminus D_v]}) \leq \beta \binom{n}{2}$ **then**
12. $\underline{\beta} = \tilde{\beta}$
13. Remove nodes from D_v as long as $\mathcal{P}(G_{[V \setminus D_v]}) \leq \beta \binom{n}{2}$
14. **if** $|D_V| > |D_v|$ **then** $D_V = D_v$
15. **else** $\overline{\beta} = \tilde{\beta}$

18.**end while**

19. Return D_V

$D'_e\}$ in $G(V, E)$ which is a vertex disruptor set in G. Since G and G' have different maximum pairwise connectivity, $\frac{(n-1)n}{2}$ for G and $\frac{(2n-1)2n}{2}$ for G', the fractions of pairwise connectivity remaining in G and G' after removing D_v and D'_e are, however, not exactly equal to each other.

In the second phase of Algorithm 3, when separating a graph into SCCs, the smaller the sizes of SCCs, the smaller pairwise connectivity in the graph. However, the smaller the maximum size of each SCC, the more edges to be cut. We perform binary search to find a right upper bound for size of each SCC in G'. In the algorithm,

the lower bound and upper bound of the size of each SCC are denoted by $\underline{\beta}|V'|$ and $\overline{\beta}|V'|$ respectively. At each step we try to find a minimum capacity edge set in $G'(V', E')$ whose removal partitions the graph into strongly connected components of size at most $\tilde{\beta}|V'|$, where $\tilde{\beta} = \lfloor \frac{\underline{\beta} + \overline{\beta}}{2\epsilon} \rfloor \times \epsilon$. We round the value of $\tilde{\beta}$ to the nearest multiple of ϵ so that the number of steps for the binary search is bounded by $\log \frac{1}{\epsilon}$. The problem of finding a minimum capacity edge set to decompose a graph of size n into strongly connected components of size at most ρn is known as ρ-separator problem. We use here the algorithm presented in [37] that for a fixed $\epsilon > 0$ finds a ρ-separator in directed graph G whose value is at most $O\left(\frac{1}{\epsilon^2}.\log n \log\log n\right)$ times $\mathrm{Opt}_{(\rho-\epsilon)}$-separator where $\mathrm{Opt}_{(\rho-\epsilon)}$-separator is the cost of the optimal $(\rho - \epsilon)$-separator. Finally, we derive the cut vertices in G from the cut edges in G' to obtain the β'-vertex disruptor.

Lemma 5. *Algorithm 3 always terminates with a β'-vertex disruptor.*

Proof. We show that whenever $\tilde{\beta} \leq \beta'$ then the corresponding D_v found in Algorithm 3 is a β'-vertex disruptor in G. Consider the edge disruptor D'_e in G' induced by D_v. We first show the mapping between SCCs in $G_{[V \setminus D_v]}$ and SCCs in $G'[E' \setminus D'_e]$, the graph obtained by removing D'_e from G'. Partition the vertex set V of G into: (1) D_v: the set of removed nodes (2) V_{single}: the set of nodes that are not in any cylcle i.e. they are SCCs of size one (3)$V_{connected}$: union of remaining SCCs that sizes are at least two, say $V_{connected} = \biguplus_{i=1}^{l} C_i, |C_i| \geq 2$. Vertices in $V_{connected}$ belong to at least one cycle in G.

We have following corresponding SCCs in $G'[E' \setminus D'_e]$:

1$v \in D_v \leftrightarrow$ SCCs $\{v^+\}$ and $\{v^-\}$. Since after removing $(v^- \rightarrow v^+)$ v^+ does not have incoming edges and v^- does not have outgoing edges.

2$v \in V_{single} \leftrightarrow$ SCCs $\{v^+\}$ and $\{v^-\}$. Since v does not lie on any cycle in G. Assume v^+ belong to some SCC of size at least 2 i.e. v^+ lies on some cycle in G'. Because the only incoming edge to v^+ is from v^-. It follows that v^- is preceding $v+$ on that cycle. Let u^-, u^+ be the successive vertices of v^+ on that cycle. We have u and v belong to a same SCC in G which yields a contradiction. Similarly, v^- cannot lie on any cycle in G'.

3SCC $C_i \subset V_{connected} \leftrightarrow$ SCC $C'_i = \{v^-, v^+ \mid v \in C_i\}$. This can be shown using a similar argument to that in the case $v \in V_{single}$.

Since D'_e is a $\tilde{\beta}$-separator, the sizes of SCCs in $G'[E' \backslash D'_e]$ are at most $\tilde{\beta}\ 2n$. It follows that the sizes of SCCs in $G_{[V \backslash D_v]}$ are bounded by $\tilde{\beta} n$. Denote the set of SCCs in $G_{[V \backslash D_v]}$ by $\mathcal{C}$ with the convention that vertices in D_v become singleton SCC in $G_{[V \backslash D_v]}$. Therefore, we have:

$$\begin{aligned}
\mathcal{P}(G_{[V \backslash D_v]}) = \sum_{C_i \in \mathcal{C}} \binom{|C_i|}{2} &= \frac{1}{2}\left(\sum_{C_i \in \mathcal{C}} |C_i|^2 - |V|\right) \\
&\leq \frac{1}{2}\left(\sum_{C_i \in \mathcal{C}} \tilde{\beta}|V|)|C_i| - |V|\right) \\
= \frac{1}{2}\left(\tilde{\beta}|V|^2 - |V|\right) \leq \tilde{\beta}\binom{|V|}{2} &< \beta'\binom{|V|}{2}
\end{aligned}$$

This guarantees that the binary search always finds a β'-vertex disruptor and completes the proof. □

Theorem 3.1. *Algorithm 3 always finds a β'-vertex disruptor whose the size is at most $O(\log n \log\log n)$ times the optimal β-vertex disruptor for $\beta'^2 > \beta > 0$.*

Proof. It follows from the Lemma 5 that Algorithm 3 terminates with a β'-vertex disruptor D_v. At some step the capacity of D_v equals to the capacity of $\tilde{\beta}$-separator D'_e in G' where $\tilde{\beta}$ is at least $\beta' - \epsilon$ according to Lemma 5 and the binary search scheme. The cost of the separator is at most $O\left(\log n \log\log n\right)$ times the $\mathrm{Opt}_{(\tilde{\beta}-\epsilon)}$-separator using the algorithm in [37].

Consider an optimal $(\beta'^2 - 9\epsilon)$-vertex disruptor D'_v of G and its corresponding edge disruptor D'_e in G'. Denote the cost of that optimal vertex disruptor by $\mathrm{Opt}_{(\beta'^2-9\epsilon)}$-VD. If there exists in $G_{[V \backslash D_v]}$ a SCC C_i so that $|C_i| > (\beta' - 2\epsilon)n$ then $\mathcal{P}(G_{[V \backslash D_v]}) > \frac{1}{2}((\beta' - 2\epsilon)n - 2)((\beta' - 2\epsilon)n - 1) > (\beta'^2 - 9\epsilon)\binom{n}{2}$ when $n > \frac{20(\beta'+1)}{\epsilon}$. Hence, every SCC in $G'_{[V \backslash D'_v]}$ have size at most $(\beta' - 2\epsilon)(2n)$ i.e. D'_e is an $(\beta' - 2\epsilon)$-separator in G'. It follows that $\mathrm{Opt}_{(\beta'^2-9\epsilon)}$-VD $\geq \mathrm{Opt}_{(\beta'-2\epsilon)}$-separator in G'.

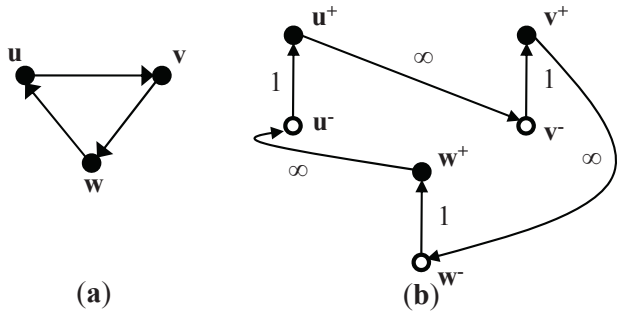


Figure 3-1. Conversion from the node version in a directed graph (a) into the edge version in a directed graph (b)

Since $\tilde{\beta} - \epsilon \geq \beta' - 2\epsilon$, we have $\mathrm{Opt}_{(\tilde{\beta}-\epsilon)}$-separator $\leq$ $\mathrm{Opt}_{(\beta'-2\epsilon)}$-separator $\leq$ $\mathrm{Opt}_{(\beta'^2-9\epsilon)}$-VD.

The size of the vertex disruptor $|D_v| = |D'_e|$ is at most $O(\log n \log\log n)$ times $\mathrm{Opt}_{(\tilde{\beta}-\epsilon)}$-separator. Thus, the size of found β'-vertex disruptor D_v is at most $O(\log n \log\log n)$ times the optimal $(\beta'^2 - 9\epsilon)$-vertex disruptor. As we can choose arbitrary small ϵ, setting $\beta = \beta'^2 - 9\epsilon$ completes the proof. □

Time complexity: Finding the separator costs $O(n^9)$ [37]. Hence, the total time complexity is $O(\log \frac{1}{\epsilon} n^9)$. However, in our experiments, the algorithm takes much less than its worst-case running time.

3.2 Connection between Edge Disruptor and Vertex Disruptor

We show that an approximation algorithm for general directed edge disruptor yields an approximation algorithm for directed vertex disruptor with (almost) the same approximation ratio.

Lemma 6. *A β-edge disruptor set in the directed graph G' induces the same cost β-vertex disruptor set in G.*

Proof. We use D_v and D'_e for vertex disruptor in G and edge disruptor in G'.

Given $\mathcal{P}(G'[E' \backslash D'_e]) \leq \beta\binom{2n}{2}$ we need to prove that: $\mathcal{P}(G_{[V \backslash D_v]}) \leq \beta\binom{n}{2}$ where $n = |V|$.

Assume $G_{[V\setminus D_v]}$ has l SCCs of size at least 2, say $C_i, i = 1 \ldots l$. The corresponding SCCs in $G'[E'\setminus D'_e]$ will be $C'_i, i = 1 \ldots l$ where $|C'_i| = 2|C_i|$.

Since $\frac{\binom{2k}{2}}{\binom{2n}{2}} - \frac{\binom{k}{2}}{\binom{n}{2}} = \frac{k(n-k)}{(n-1)n(2n-1)} \geq 0$, for all $0 \leq k \leq n$. We have

$$\frac{\mathcal{P}(G_{[V\setminus D_v]})}{\binom{n}{2}} = \sum_{i=1}^{l} \frac{\binom{|C_i|}{2}}{\binom{n}{2}} \leq \sum_{i=1}^{l} \frac{\binom{|C'_i|}{2}}{\binom{2n}{2}} \leq \beta$$ □

Lemma 7. *A β-vertex disruptor set in G induces the same cost $(\beta+\epsilon)$-edge disruptor set in G' for any $\epsilon > 0$.*

Proof. We use the same notations in the proof of Lemma 6. Given $\mathcal{P}(G_{[V\setminus D_v]}) \leq \beta\binom{n}{2}$ we need to prove that: $\mathcal{P}(G'[E'\setminus D'_e]) \leq (\beta+\epsilon)\binom{2n}{2}$. We have:

$$\begin{aligned}
& \frac{\mathcal{P}(G'[E'\setminus D'_e])}{\binom{2n}{2}} \\
= & \sum_{i=1}^{l} \frac{|C_i|(n-|C_i|)}{(n-1)n(2n-1)} + \frac{\mathcal{P}(G_{[V\setminus D_v]})}{\binom{n}{2}} \\
= & \frac{\mathcal{P}(G_{[V\setminus D_v]})}{\binom{n}{2}} \left(1 - \frac{1}{2n-1}\right) + \frac{\sum_{i=1}^{l}|C_i|}{n(2n-1)} \\
< & \beta + \frac{1}{2n-1} < \beta + \epsilon
\end{aligned} \tag{3–1}$$

when $n \geq \lfloor \frac{1+\epsilon}{2\epsilon} \rfloor + 1$. □

Theorem 3.2. *Given a factor $f(n)$ polynomial time approximation algorithm for β-edge disruptor, there exists a factor $(1+\epsilon)f(n)$ polynomial time approximation algorithm for β-vertex disruptor where $\epsilon > 0$ is an arbitrary small constant.*

Proof. Let G be a directed graph with uniform vertex costs in which we wish to find a β-vertex disruptor. Construct G' as described at the beginning of this Section.

Apply the given approximation algorithm to find in G' a β-edge disruptor, denoted by D'_e, with the cost at most $f(n)\cdot \mathrm{Opt}_{\beta-\mathrm{ED}}(G')$, where $\mathrm{Opt}_{\beta-\mathrm{ED}}(G')$ is the cost of a minimum β-edge disruptor in G'. From Lemma 6, D'_e induces in G a β-vertex disruptor D_v of the same cost. We shall prove that

$$\mathrm{Opt}_{\beta-\mathrm{ED}}(G') \leq \mathrm{Opt}_{\beta-\mathrm{VD}}(G) + \gamma_0,$$

where $\mathrm{Opt}_{\beta-\mathrm{VD}}(G)$ is the cost of a minimum β-vertex disruptor in G and γ_0 is some positive constant. It follows that the cost of D_v will be at most

$$f(n) \cdot (\mathrm{Opt}_{\beta-\mathrm{VD}}(G) + \lambda_0) \leq (1+\epsilon) f(n) \mathrm{Opt}_{\beta-\mathrm{VD}}(G)$$

Here, we assume that $\mathrm{Opt}_{\beta-\mathrm{VD}}(G) > \frac{\gamma_0}{\epsilon}$ otherwise we can find $\mathrm{Opt}_{\beta-\mathrm{VD}}(G)$ in time $O(n^{\frac{\gamma_0}{\epsilon}+2})$.

From an optimal β-vertex disruptor of G, construct its corresponding edge disruptor D_e^* in G'. If $\mathcal{P}(G'[E \setminus D_e^*] \leq \beta\binom{2n}{2}$ then $\mathrm{Opt}_{\beta-\mathrm{ED}}(G') \leq \mathrm{Opt}_{\beta-\mathrm{VD}}(G)$ and we yield the proof. Thus, we consider the case $\mathcal{P}(G'[E \setminus D_e^*] > \beta\binom{2n}{2}$.

Among SCCs of $G'[E \setminus D_e^*]$, there must be a SCC of size at least $\beta 2n$ or else $G'[E \setminus D_e^*] \leq \beta^{-1}\binom{\beta 2n}{2} \leq \beta\binom{2n}{2}$ (contradiction). Remove $\gamma_0 = \left\lceil \frac{1}{\beta} \right\rceil$ vertices from that SCC. The pairwise connectivity in $G'[E \setminus D_e^*]$ will decrease at least $(\beta 2n - \frac{1}{\beta})\frac{1}{\beta} = 2n - \frac{1}{\beta^2} \geq n$ for sufficient large n. From Eq. 3–1 in Lemma 7, the pairwise connectivity after removing vertices will be less than

$$(\beta + \frac{1}{2n-1})\binom{2n}{2} - n \leq \beta\binom{2n}{2}$$

Therefore, after removing at most γ_0 vertices from D_e^*, we get a β-edge disruptor. Hence,

$$\mathrm{Opt}_{\beta-\mathrm{ED}}(G') \leq \mathrm{Opt}_{\beta-\mathrm{VD}}(G) + \gamma_0.$$ □

3.3 Branch-and-cut Algorithm

Branch-and-cut methods have proven to be a very successful approach for solving a wide variety of integer programming problems. In contrast with meta-heuristics, they can guarantee optimality. They combine a *branch-and-bound* algorithm with a *cutting plane method* that is used to improve the solution of the linear programming relaxations.

This section presents components of our branch-and-cut algorithm. We begin with a new lightweight mixed integer programming formulation for β-vertex disruptor in Subsection 3.3.2. In the next subsection, we introduce a new class of strong cutting

planes and the separation procedure to find such cutting planes. The primal heuristics that provides upper bounds for pruning during the search process is presented in Subsection 3.3.4.

3.3.1 Mixed Integer Programming Formulation

We model the network as an undirected graph $G = (V, E)$ of n nodes numbered from 1 to n; the degree of node $1 \leq i \leq n$ is denoted by $d(i)$. The pairwise connectivity of G, denoted by $\mathcal{P}(G)$ is the number of node pairs with at least one path between them. For example, if G is connected, then $\mathcal{P}(G) = \binom{n}{2}$.

Given a positive constant $0 \leq \beta \leq 1$, a subset of vertices $S \subset V$ in G is a *β-vertex disruptor* if the subgraph $G_{[V \setminus S]}$, induced by $V \setminus S$ in G, has pairwise connectivity at most $\beta\binom{n}{2}$. The β-edge disruptor problem asks to find a β-vertex disruptor of the minimum size.

The problem can be generalized so that each node $u \in V$ has a cost $w(u)$ of removing and we wish to find a β-vertex disruptor of the minimum cost. This generalization is straightforward and shall be ignored to simplify the presentation.

The IP formula for β-vertex disruptor (IP_{vd}) is as follow

$$\text{minimize} \quad \sum_{i=1}^{n} s_i \tag{3–2}$$

$$\text{subject to } d_{ij} \leq s_i + s_j, \qquad (i,j) \in E, \tag{3–3}$$

$$d_{ij} + d_{jk} \geq d_{ik}, \qquad \forall i \neq j \neq k \tag{3–4}$$

$$\sum_{i<j} d_{ij} \geq (1-\beta)\binom{n}{2}, \tag{3–5}$$

$$s_i \leq d_{ij}, \qquad i \neq j \tag{3–6}$$

$$s_i, d_{ij} \in \{0,1\}, \qquad i,j \in [1..n] \tag{3–7}$$

We use variable d_{ij} to represent the "distance" between a pair of nodes i and j in the residual network i.e.

$$d_{ij} = \begin{cases} 0 & \text{if } i \text{ and } j \text{ are in the same connected component} \\ 1 & \text{otherwise.} \end{cases}$$

An extra variable s_i is used for each node $i \in V$, where

$$s_i = \begin{cases} 0 & \text{if node } i \text{ is not removed} \\ 1 & \text{if } i \text{ is removed (selected into the disruptor.)} \end{cases}$$

The objective minimizes the total number of removed nodes i.e. the size of the vertex disruptor. Note that $d_{ij} = d_{ji} \quad \forall (i,j) \in V \times V$. Constraint (4–3) is the well-known *triangle inequality* which implies that if i and j are connected, and j and k are connected, then i and k must be connected. Constraint (4–4) limits the pairwise connectivity in G to be at most $\beta\binom{n}{2}$.

Constraint (4–2) implies the base case that if i and j are neighbors and neither i or j is removed ($s_i = s_j = 0$), then i and j remain connected i.e. $d_{ij} = 0$. Constraint (4–5) states the fact that a removed node will not connect to any other nodes ($s_i = 1 \rightarrow d_{ij} = 1$).

There are several drawbacks with the IP formula of the β-vertex disruptor problem (IP_{vd}) (and also formulations of k-CND and k-CED [10]). A large number of integral variables, $\Theta(n^2)$, makes the selection of branching difficult and significantly increases the depth and size of the search tree. In addition, excessive number of constraints, $\Theta(n^3)$, even for small sized instances leads to a large linear programming relaxation that consumes an extremely large amount of memory and computing time.

3.3.2 Sparse Metric Technique

We first devise a new Mixed-Integer Programming (MIP) formulation for the β-vertex disruptor problem that consists of only n integer variables and much smaller number of constraints. Since the only role of triangle inequalities is to guarantee d_{ij} to be a

pseudo-metric (as defined later in the proof of Theorem 3.3), we introduce a compact subset of inequalities, so-called *sparse metric*, that guarantees the same pseudo-metric property. When the network is sparse i.e. $|E| \propto |V|$, the number of constraints reduces substantially from $\Theta(n^3)$ to $\Theta(n^2)$.

Our new MIP formulation for the β-vertex disruptor problem (MIP$_{\text{vd}}$) is similar to IP$_{\text{vd}}$ except in places of constraints (4–3) and (5–9) as presented below.

$$d_{ij} + d_{jk} \geq d_{ik}, \qquad k \in N_{\min}(i,j) \tag{3–8}$$

$$d_{ij} \in [0,1], \qquad i,j \in [1..n], \tag{3–9}$$

where $N_{\min}(i,j)$ is the set of neighbors of i excluding j if $d(i) < d(j)$, and $N_{\min}(i,j)$ is the set of neighbors of j excluding i, otherwise. We also drop the integral requirements on d_{ij} i.e. replace the constraints $d_{ij} \in \{0,1\}$ with $d_{ij} \in [0,1]$. However, the integrality of $s_1, s_2, \ldots, s_n$ remains.

Note that there are exactly n integer variables $s_1, s_2, \ldots, s_n$. In addition, the number of constraints is upper bounded by

$$\begin{aligned} &|E| + \sum_{i<j} \min\{d(i), d(j)\} + \frac{n(n-1)}{2} \\ \leq &n^2 + \sum_{i<j} \frac{d(i)+d(j)}{2} = n^2 + \frac{n-1}{2} \sum_{i=1}^{n} d(i) = O(mn) \end{aligned}$$

Hence, the number constraints is substantially less than $O(n^3)$ for complex networks that are often sparse.

We proceed to prove the equivalence of the compact formulation MIP$_{\text{vd}}$ to IP$_{\text{vd}}$ by showing the following

- The integrality constraints on d_{ij} $\forall i,j$ are in fact redundant (Proposition 3.1).
- The optimal solutions of MIP$_{\text{vd}}$ also induce optimal β-vertex disruptor in G (Theorem 3.3).

- The optimal fractional solution of LP relaxation of IP_{vd} can be found by solving the (smaller) LP relaxation of MIP_{vd}, following by an $O(mn + n^2 \log n)$ tuning procedure (Theorem 3.4).

Proposition 3.1. *For every optimal solution of MIP_{vd}, there is a feasible solution of the MIP with the same objective value in which all variables are integral.*

Proof. Round all $d_{ij} > 0$ to 1. This will not violate constraints (4–5) and (4–4). For constraints (4–2), if d_{ij} is rounded up to 1 then the integrality of s_i, s_j implies $s_i + s_j \geq 1$, or else if $d_{ij} = 0$ then the constraints are still satisfied. Assume the rounding violates constraints (3–8) for some triple (i, j, k). This happens if and only if $d_{ik} = 1$ and $d_{ij} = d_{jk} = 0$. Hence, before rounding, $d_{ik} > 0$ and $d_{ij} = d_{j,k} = 0$ that contradicts the constraint $d_{ij} + d_{jk} \geq d_{ik}$. It follows that rounding gives a feasible integral solution to the MIP. □

Let $\mathcal{D}_{\text{MIP}} = \{\, i \mid s_i = 1\}$ be the disruptor induced by the optimal solution of MIP_{vd} and $\text{OPT}^{\beta}_{\text{vd}}$ be an optimal β-vertex disruptor.

By setting $s_i = 0\ \forall i \in \text{OPT}^{\beta}_{\text{vd}}$ and $d_{ij} = 0$ for all i, j in a same connected component of $G_{[V \setminus \text{OPT}^{\beta}_{\text{vd}}]}$ and $d_{ij} = 1$ if not, we yield a feasible solution for MIP_{vd}. Therefore,

$$|\mathcal{D}_{\text{MIP}}| \leq |\text{OPT}^{\beta}_{\text{vd}}|$$

Theorem 3.3. *The optimal solution $\mathcal{D}_{\text{MIP}} = \{\, i \mid s_i = 1\}$ obtained by solving MIP_{vd} is a minimum β-vertex disruptor of G.*

Proof. Since $|\mathcal{D}_{\text{MIP}}| \leq |\text{OPT}^{\beta}_{\text{vd}}|$, we only need to show that $\mathcal{D}_{\text{MIP}}$ is a β-vertex disruptor.

Assume that we can prove that $d_{ij} = 0$ for every connected pairs (i, j) in $G_{[V \setminus \mathcal{D}_{\text{MIP}}]}$. Then, only disconnected pairs $d_{i'j'}$ will contribute to the sum in constraint (4–4). Since $d_{i'j'} \leq 1 \quad \forall i, j \in [1..n]$, the number of disconnected pairs must be at least $(1 - \beta)\binom{n}{2}$. It will follow that $\mathcal{D}_{\text{MIP}}$ is a β-vertex disruptor.

Hence, the rest of the proof is to show that $d_{ij} = 0$ for every connected pairs (i, j) in $G_{[V \setminus \mathcal{D}_{\text{MIP}}]}$.

Note that d is a pseudo-metric, i.e., the function $d(i,j) = d_{ij}$ satisfy:

- Non-negativity: $d(i,j) \geq 0$
- Identity: $d(i,i) = 0$
- Symmetry: $d(i,j) = d(j,i)$
- Subadditivity: $d(i,j) \leq d(i,k) + c(k,j)$.

For each connected pair (i,j) in $G_{[V \setminus \mathcal{D}_{\text{MIP}}]}$, we prove that $d_{ij} = 0$ by induction on the length t of the shortest path (in number of hops) between nodes i and j.

The basis. The statement holds for $t = 1$. By constraint (4–2), if $(i,j) \in E$ and i, j are connected in G i.e. $s_i = s_j = 0$, then $d_{ij} \leq s_i + s_j = 0$. Since $d_{ij} \geq 0$, it follows that $d_{ij} = 0$.

The inductive step. Assume that the statement holds for $t = t'$, we show that the statement is also true for $t = t' + 1$. Let i, j be some pairs connected with a path of length at most $t' + 1$. Since removing all nodes in $N_{\min}(i,j)$ disconnects i from j, *the path between* i *and* j *must pass through some node* $k \in N_{\min}(i,j)$. In addition, the shortest paths from i to k and from k to j have lengths at most t'. Thus, by the induction hypothesis we have $d_{ik} = d_{kj} = 0$. It follows from the constraint in (3–8) that $d_{ij} \leq d_{ik} + d_{kj} = 0$. Thus, the statement holds for all $t > 0$. □

Finally, we show the relationship between the LP relaxation of IP_{vd} and that of MIP_{vd}.

Theorem 3.4. *The optimal solution of the LP relaxation* IP_{vd} *can be found by solving the LP relaxation of* MIP_{vd}*, following by an* $O(mn + n^2 \log n)$ *tuning procedure.*

Proof. Let (s,d) be an optimal fraction solution of the LP relaxation of MIP_{vd}. Associate a weight d_{ij} for each edge $(i,j) \in E$. Let d'_{ij} be the shortest distance between two nodes (i,j) with the new edge weights. We have

- $d'_{ij} \geq d_{ij}$ for all i, j and $d'_{ij} = d_{ij} \forall (i,j) \in E$.
- $d'_{ij} = \min_{k=1}^{n} \{d'_{ik} + d'_{kj}\}$. Hence, d'_{ij} is a pseudo-metric.

The first statement can be shown by the same induction in the proof of Theorem 1. The second statement comes from the definition of d'_{ij}.

Furthermore, we define $d^*_{ij} = \min\{d'_{ij}, 1\}$. If we use the Johnson's algorithm [27] to compute all pairs shortest paths d'_{ij}, the time complexity to construct d^*_{ij} from d_{ij} is $O(mn + n^2 \log n)$. We shall prove that (s, d^*) is a feasible solution of IP_{vd} by showing that (s, d^*) satisfies all constraints in IP_{vd}.

By definition, we have $d^*_{ij} = \min\{d'_{ij}, 1\} \geq \min\{d_{ij}, 1\} = d_{ij}\ \forall i, j$ and $d^*_{ij} = d_{ij}\ \forall (i, j) \in E$. Thus, for all $(i, j) \in E$, $d^*_{ij} = d_{ij} \leq s_i + s_j$.

In addition, d^* is also a pseudo-metric as $d^*_{ik} + d^*_{kj} \geq \min\{d'_{ik} + d'_{kj}, 1\} \geq \min\{d'_{ij}, 1\} = d^*_{ij}$. From $d^*_{ij} \geq d_{ij}$, we have $\sum_{i,j} d^*_{i,j} \geq \sum_{i,j} d_{i,j} \geq \beta\binom{n}{2}$ and $s_i \leq d_{ij} \leq d^*_{ij}$. Thus, (s, d^*) is a feasible solution of IP_{vd}.

Obviously, the minimum objective of the LP relaxation of MIP_{vd} is smaller or equal to that of IP_{vd}. Since, the objective values associate with (s, d^*) and (s, d), a minimum solution of the LP relaxation of MIP_{vd}, are the same, (s, d^*) must be a minimum solution of the LP relaxation of IP_{vd}. □

3.3.3 Cutting Planes

We present a class of strong cutting planes together with the separation procedure to identify those cutting planes. These can be used in conjunction with cutting planes generated automatically by optimization packages to improve the convergence of the branch-and-cut algorithm.

3.3.3.1 Vertex-Connectivity and Invalid Inequalities

One often overlooked characteristic of solutions for clustering and partitioning problems on graph is that clusters must induce connected subgraph. This characteristic is not reflected in either IP_{vd} or MIP_{vd} formulations.

A subset $S \subset V$ is a *vertex-cut* for a pair (u, v), if removing S from graph G, disconnect s and t. For all vertex-cut S of (u, v), if $\sum_{i \in S} s_i = |S|$, then d_{uv} must be one.

Thus, we have VC inequality

$$\sum_{i \in S} s_i - d_{uv} \leq |S| - 1$$

This inequality is valid for all feasible points inside the polyhedra of MIP_{vd}.

Algorithm 6. Separation procedure for VC inequalities

for each pair $(u, v) \in V \times V$ **do**
  Construct a flow network $G = (V, E)$ as follows
    Assign u and v as source and sink, respectively
    Each node $k \in V$ has capacity $\bar{s}_k$
    Every edge has capacity ∞
    if $(u, v) \in E$, **then** (u, v) has capacity zero.
  Find the maximum-flow (min-cut)
  if maximum-flow is less than $\bar{d}_{uv}$, **then**
    Find the min vertex-cut S
    Add the VC inequality associated with S to MIP
  end if
end for

3.3.3.2 Separation Procedure for VC Inequalities

Given a point (fractional solution) $(s, d) \in \mathbb{R}^{\binom{n+1}{2}}$, an exact separation algorithm for some class of inequalities either finds a member of the class violated by (s, d), or proves that no such member exists. In many cases, finding such algorithm is intractable (NP-hard problem) and one has to settle for heuristic procedures. Fortunately, there is an exact algorithm for our separation procedure based on finding the max-flow on the network with node capacities.

The VC inequality can be rewritten as

$$\sum_{i \in S} \bar{s}_i - \bar{d}_{uv} \geq 0, \quad \text{S is any vertex-cut of } (u, v)$$

where $\bar{s}_i = 1 - s_i$ and $\bar{d}_{uv} = 1 - d_{uv}$.

Algorithm 7. Sharpest Decreasing Vertices (SDV)

Start with some β-vertex disruptor $\mathcal{D} \subseteq V$.
Repeat
  while (true) **do**
    $u = \arg\min_{v \in \mathcal{D}} \{\mathcal{P}(G_{[V \setminus (\mathcal{D} \cup \{v\})]})\}$
    if ($\mathcal{D} \setminus \{u\}$ is a β-disruptor) **then** $\mathcal{D} = \mathcal{D} \setminus \{u\}$
  for each ($w \in V \setminus \mathcal{D}$) **do**
    $D_w = \mathcal{D} \cup \{w\}$
    $u = \arg\min_{v \in \mathcal{D}} \{\mathcal{P}(G_{[V \setminus (D_w \cup \{v\})]})\}$
    +**if** ($u \neq w$) **then** $\mathcal{D} = D_w \cup \{u\}$
Until ($\mathcal{D}$ not changing)
Output $\mathcal{D}$.

Therefore, the point (s, d) violates this inequality if and only if $\sum_{i \in S} \bar{s}_i < \bar{d}_{uv}$. The most violated inequality is the one that minimize the sum $\sum_{i \in S} \bar{s}_i$, given S is a vertex-cut of (u, v). Thus, the subset S corresponding to the most violated inequalities can be found using minimum capacitated vertex-cut of (u, v). The separation procedure is described in Algorithm 1. Here, we need to solve the maximum-flow (min-cut) problem in networks with both node and edge capacities. If we apply Push-relabel algorithm with dynamic trees [42], the time complexity to find cutting planes for one node pair is $O(mn \log \frac{n^2}{m})$. The total time complexity for the separation procedure will be $O(n^3 m \log \frac{n^2}{m})$. In our implementation, this procedure is called sparingly in order to avoid excessive running time.

3.3.4 Primal Heuristic

The search for an optimal solution in a branch and cut algorithm can be accelerated by obtaining a high quality feasible solution to provide upper bounds for pruning other subproblems. We present a heuristic that rounds the fractional solution of MIP relaxations to get integral solutions.

Let (s, d) be a fractional solution of an LP relaxation. We first sort s_i in non-decreasing order $s_{i_1} \leq s_{i_2} \leq \ldots \leq s_{i_n}$. Then we round down all $s_{i_1}, s_{i_2}, \ldots, s_{i_k}$ to zero and round

up $s_{i_{k+1}}, s_{i_{k+2}}, \ldots, s_{i_n}$ to one, where k runs from 1 to n. If the obtain solution is a β-vertex disruptor, a local search method described in Algorithm 2 is then used to refine the solution. The local search method refines the solution by repeatedly:

- Removing node(s) from the disruptor if possible
- Swapping a node w outside the disruptor with a node u in the disruptor that gives the sharpest decrease in connectivity.

The local search terminates when no improvement exists.

3.4 Experimental study

We perform experiments to find out the gap between the solution of the pseudo approximation algorithm (Algorithm 3) and an optimal solution found by solving an Integer programming formulation. We generate two types of network: random networks following Erdos-Rényi model and power-law networks following Barabási-Albert model. For each type of network, we generate different instances with number of nodes ranging from 30 to 100. Edge densities of generated networks are around 10%. The machine used for the experiments was an 8 cores 2.2 Ghz equipped with 64 GB memory.

Size of disruptors found by Algorithm 3 and the size of optimal disruptors are presented in Tables 3-1 and 3-2. Despite a large theoretical gap of the pseudo approximation algorithm, the algorithm produces near-optimal solutions and returning optimal solutions in more than half places (marked with bold numbers).

Especially, our algorithm performs extremely well on power-law networks. It misses the optimal solution in only one place when the number of vertices is 90. Between a random network and a power-law network of roughly same sizes, the size of disruptor in the power-law network is significantly smaller (approximately 50%) than that in the random network, showing extremely high degree of vulnerability of power-law network to attacks [7].

Table 3-1. Size of disruptor on Erdos-Rényi networks at 60% connectivity.

Vertex	30	40	50	60	70	80	90	100
Edge	43	78	122	177	241	316	400	495
Optimal	2	4	7	9	11	12	16	18
Approx	3	**4**	8	**9**	**11**	13	**16**	19

Table 3-2. Size of disruptor on Barabási–Albert networks at 60% connectivity.

Vertex	30	40	50	60	70	80	90	100
Edge	54	131	189	208	245	262	354	445
Optimal	1	3	5	6	6	5	7	9
Approx	**1**	**3**	**5**	**6**	**6**	**5**	10	**9**

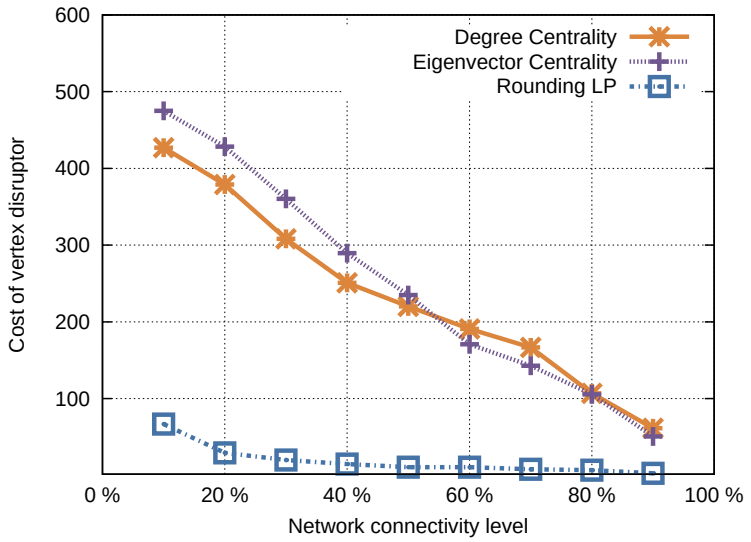


Figure 3-2. Disruptors found by different methods in the Western States Power Grid of the United States at different levels of disruption.

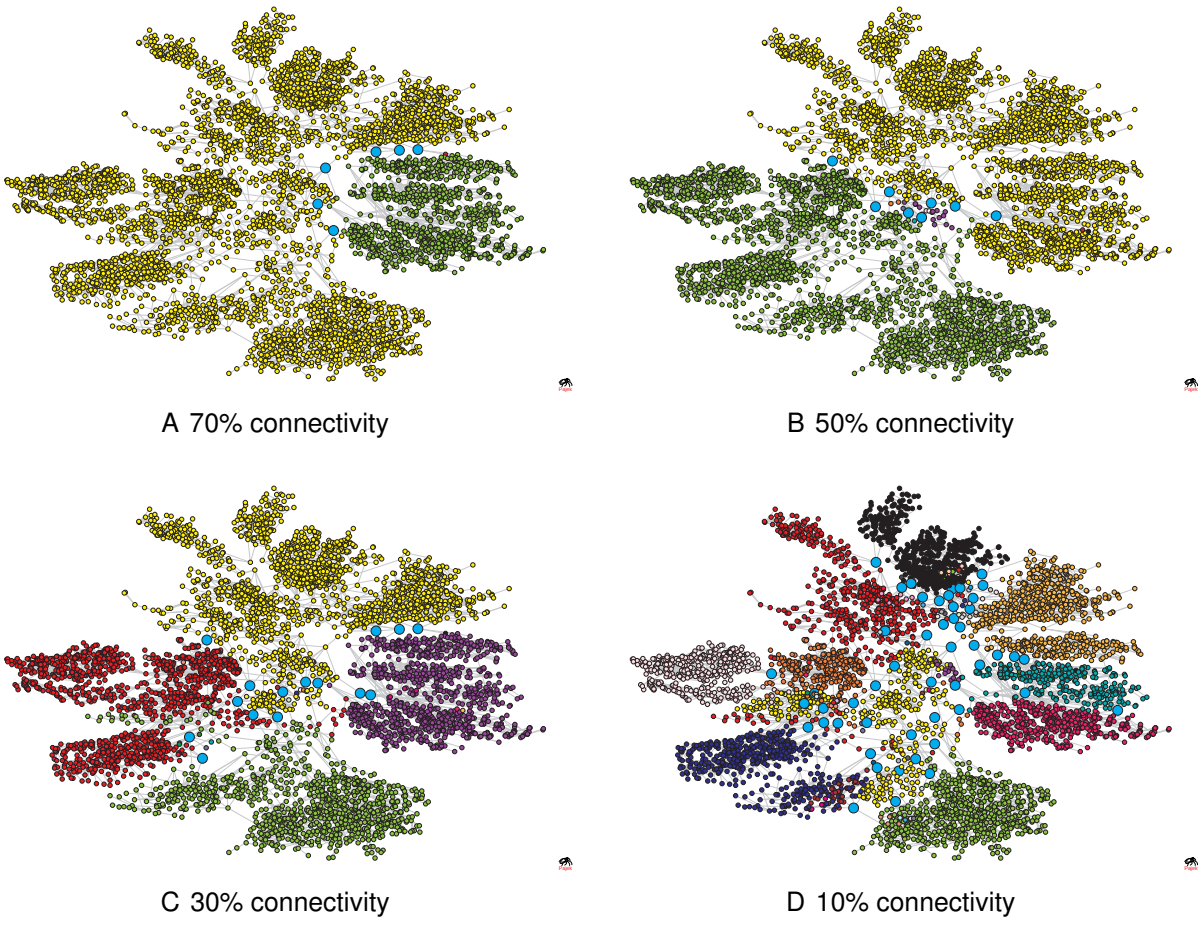


Figure 3-3. Disruptors found by different methods in the Western States Power Grid of the United States at different levels of disruption.

The running time for solving the Integer programming increases from few minutes to 10 hours for the largest test cases, while in the longest run, the pseudo-approximation algorithm takes only 29 seconds.

3.4.1 Performance of the Branch-and-cut Algorithm

Vertex	Edge	β	Removed vertex	Time (seconds)		Constraint	
				IP_{vd}	MIP_{vd}	IP_{vd}	MIP_{vd}
50	141	60.0%	4	63	8	60, 167	4, 861
150	286	1.0%	18	19, 788	2	1, 665, 362	31, 887
-	-	5.0%	15	18, 070	7	-	32, 161
-	-	8.0%	12	n/a	73	-	33, 242
-	-	10.0%	11	n/a	1, 363	-	39, 615
-	-	20.0%	9	n/a	1, 737	-	39, 313
-	-	40.0%	7	n/a	2, 149	-	42, 830
-	-	60.0%	5	n/a	1, 610	-	38, 458
-	-	90.0%	2	26, 277	147	-	34, 321
200	387	60.0%	8	n/a	64, 860	3, 960, 488	72, 980
600	1, 166	0.5%	69	n/a	48, 918	107, 641, 467	516, 656
1000	1, 959	0.5%	198	n/a	747	499, 340, 027	1, 437, 326

Table 3-3. Comparisons of IP_{vd} and MIP_{vd} on power-law networks

We implement our branch and cut algorithm using GUROBI 4.0 on a computer with Intel Xeon 2.93 Ghz processor and 12 GB memory. Table 3-3 shows results for IP_{vd} and our new branch and cut algorithm (MIP_{vd}) on power-law networks [12] of various sizes. We report for each disruption level β, the number of removed vertices in the optimal solution, the number of Rows (constraints), Nonzeros (nonzero coefficients), and solving time.

As shown in Table 3-3, our branch-and-cut algorithm utilizing sparse metric technique and strong cutting planes is substantially faster and more memory-efficient than the original branch-and-cut equipped in GUROBI MIP solver. The speed up factor is from 8 times for 50 nodes to several thousand times for larger instances. For the

network of 150 nodes, MIP_{vd} often takes less than 30 minutes, while IP_{vd} runs out of memory or does not terminate after 100,000 seconds (noted with *n/a*).

3.4.2 Case study: Western States Power Grid

We study a network of 4941 nodes and 6594 edges representing the topology of the Western States Power Grid of the United States. The network is shown to be high clustering with small characteristics path lengths [79]; hence the network is rather vulnerable to targeted attacks.

It is intractable to find the optimal disruptor using Integer Programming for such a large network. Our approximation algorithm uses row-generation technique to reduce excessive amount of constraints and runs on a clusters of 20 nodes, each node is equipped with an 8 cores 2.2 Ghz CPU and 64 GB memory.

We compare the attack schemes that target nodes based on their centrality with our pseudo approximation algorithm to show that those methods might not be suitable to reveal network vulnerability in term of overall network connectivity. Compared methods include

1. *Degree Centrality*: The algorithm sequentially remove node with the maximum degree until the pairwise connectivity in the graph less than $\beta\binom{n}{2}$.
2. *Betweenness Centrality*: We repeatedly remove the node with maximum betweenness centrality, until the pairwise connectivity in the graph less than $\beta\binom{n}{2}$. Recall that the betweenness $Bt(v)$ for node v is: $Bt(v) = \sum_{\substack{s\neq v\neq t\in V\\ s\neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$ where σ_{st} is the number of shortest paths from s to t, and $\sigma_{st}(v)$ is the number of shortest paths from s to t that pass through a node v.
3. *Eigenvector Centrality*: Nodes are removed in descending order of their Eigenvector centrality (Pagerank) values with the default damping factor of 85% as in [65].

We show in Figure 3.4 vulnerability reported by different methods at various levels of disruption. The network is surprisingly vulnerable to targeted attacks. For example to reduce 40% connectivity in the network (60% connectivity remain) we only need to destroy 0.16% stations. Bringing down the connectivity to the same level, the average number of nodes to remove for random networks and power-law networks are 13%

and 3% respectively. Even destroying only 1% of stations can dramatically disrupt 90% connectivity in the network.

None of other methods can reveal correctly the vulnerability of the power grid. Their disruptor sizes are 6 to 20 times larger than those of our approximation algorithm. Thus, using alternative assessment methods rather than the ones we proposed might lead to a dangerous mirage that the network is strongly stable.

Because of high clustering property, nodes that lie among clusters in the networks will often have high betweenness values. Intuitively, we expected the betweenness method to easily identify those nodes and perform well in the experiment. Surprisingly, the performance of betweenness method turns out to be even worse than that of degree centrality.

We visualize the network fragmentation at varied disruptive levels in Figures 3-3C and 3-3D. Disruptor separates the network into large connected components. We observe that not all nodes in the disruptor at the 30% connectivity level are in the disruptor at 10% level. Hence, we cannot assume nodes in the disruptor at the lower levels is the superset of nodes in that at the higher level. It explains why centrality assessment methods in which nodes are selected in a fixed order fail to exhibit the vulnerability of the network at different disruptive levels.

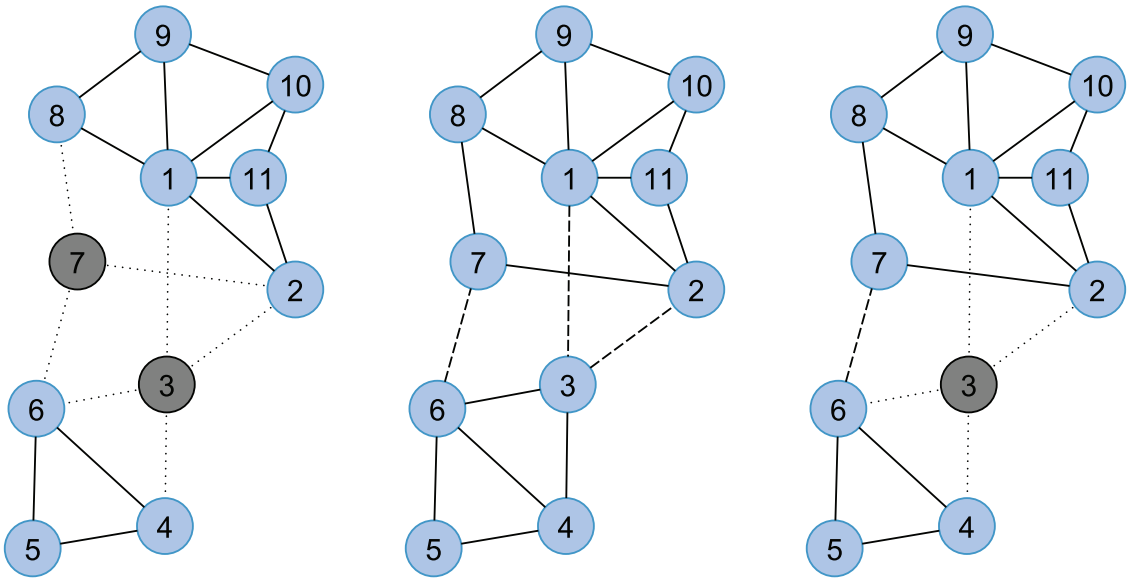


Figure 4-1. A) After removing nodes 1 and 2 with highest degree, the network remains connected and the pairwise connectivity reduces only 35%. As shown in **a.**, the solution that minimizes the connectivity (nodes 3 and 7) effectively breaks the network into two parts, disrupting 67% connectivity. B) Minimum cost solutions to reduce 50% of the connectivity assuming links have cost 2 and nodes have cost 3 **a.** node only & **b.** link only **c.** joint nodes & links. The minimum cost is 6 if attacking only nodes or only links, and is 5 if both links and nodes are targeted. Thus, it is insufficient to study node and link attacks separately.

We begin with a network sample that show the advantage of the pairwise connectivity metric over the node centrality measures in Fig. 4-1A. Assume two nodes are to be removed from our simple example. If the two nodes are selected according to their degree centrality, nodes 1 and 2 will be removed and the network remains connected. However, if we remove nodes to minimize the pairwise connectivity, nodes 3 and 7 are going to be targeted, and the network is effectively broken into two smaller components. The fraction of pairwise connectivity in the residual network, denoted by β, reduces

drastically to $\beta = \frac{18}{55} \approx 33\%$. Apparently, optimizing the pairwise-connectivity metric reveals more accurate insights on the network vulnerability.

Fig. 4-1 also illustrates a fundamental shortcoming of existing work: the ability to assess network vulnerability under *joint node and link attacks*. The three sub-figures show the minimum cost attack strategies to reduce $\beta = 50\%$ pairwise connectivity, assuming each link has cost 2 and each node has cost 3. While the minimum costs for both node-attack (Fig. 4-1A) and link-attack (Fig. 4-1B) are 6, the minimum cost for node-link attacks (node 3 and link $(6, 7)$) (Fig. 4-1C) is only 5. Thus, it is insufficient to assess link vulnerability and node vulnerability separately when both links and nodes in the network can be targeted. To make matters worse, assume node 3 and link $(6, 7)$ have the same cost $\epsilon > 0$, the minimum costs for node, link, and node-link attacks will be $3 + \epsilon, 4 + \epsilon$, and 2ϵ, respectively. As the ratios $(3 + \epsilon)/(2\epsilon)$ and $(4 + \epsilon)/(2\epsilon)$ go unbounded, the existing methods can seriously misjudge the network vulnerability.

To address the shortcoming, we study the effect of *joint node and link attacks* in term of connectivity. We introduce a new problem, called β-disruptor, that finds a minimum cost set of *nodes and links* whose removal degrades the pairwise connectivity to a great extent (a fraction β). The β-disruptor problem aims to provide a more comprehensive assessment on network vulnerability. It generalizes both the β-vertex disruptor and the β-edge disruptor problems proposed in our previous work [33]. To our best knowledge, this is the first work to address the effect of simultaneous attacks on both links and nodes on network connectivity.

Our contributions are summarized as follows

- Providing an underlying framework toward assessing vulnerability under *joint node and link attacks* and formulating it as an optimization problem β-disruptor. Other performance measures such as the maximum flow between a given source-destination pair, the average maximum flow between pairs of nodes, etc. can also be used in place of pairwise connectivity to define new problems.
- Our major result is an $O\left(\sqrt{\log n}\right)$ bicriteria approximation algorithm for both *undirected and directed* networks. The algorithm finds a β-disruptor with the cost

at most $O\left(\sqrt{\log n}\right)$ times that of an optimal $\beta\prime$-disruptor, with $\beta\prime$ slightly less than β. We propose an efficient heuristic using recursive spectral bisection and variable neighborhood search. Finally, our experiments on both synthetic and real-world datasets indicate the efficacy and scalability of our proposed algorithms.

We briefly present terminologies and problem definitions in Section 4.1. Then we propose the $O(\sqrt{\log n})$ bicriteria approximation algorithm for β-disruptor in Section 4.2. Section 4.3 presents the efficient heuristic to find β-disruptor. We obtain numerical results for the presented algorithms in Section 4.4.

4.1 Mixed Removal of Nodes and Links

Once again, we abstract our general network model as a graph $G = (V, E)$, where V refers to a set of nodes and E refers to a set of links. Each vertex $u \in V$ is associated with a cost $c(u) \leq 0$ and each edge $(u, v) \in E$ has a cost $c(u, v) \geq 0$. For convenience, we also denote the number of nodes and links by n and m, respectively.

If G is an undirected graph, a vertex pair $(u, v) \in V \times V$ is connected iff there exists a path between u and v. If G is a directed graph, a vertex pair (u, v) is said to be connected if there exist paths between u and v in *both directions*. We denote the pairwise connectivity of a graph G by $\mathcal{P}(G)$. Apparently, the pairwise connectivity is maximized at $\binom{n}{2}$ when G is a (strongly) connected graph. For convenience, we use the word component to refer to connected component in undirected graphs and *strongly connected component* (SCC) in directed graphs whenever the context is clear.

β-disruptor. Given $0 \leq \beta \leq 1$, a β-disruptor is a pair of subsets

$$D_\beta = (V_\beta \subseteq V, E_\beta \subseteq E)$$

that removal from G will make the pairwise connectivity in the residual graph $G\prime = (V \setminus V_\beta, E \setminus (E_\beta \cup V_\beta \times V_\beta))$ to be at most $\beta\binom{n}{2}$. The *β-disruptor problem* asks for a β-disruptor with the minimum total cost

$$c(D_\beta) = \sum_{u \in V_\beta} c(u) + \sum_{e \in E_\beta} c(e).$$

There are two special types of β-disruptor: if $V_\beta = \emptyset$, then D_β is a *β-edge disruptor*; and if $E_\beta = \emptyset$, then D_β is a *β-vertex disruptor*. The uniform-cost versions of *β-edge disruptor problem* and the β-vertex disruptor problem are previously studied in [32].

4.1.1 Mixed Integer Linear Programming

The β-disruptor can be formulated as an Mixed Integer Linear Programming (MILP) problem as follows

$$\text{minimize} \quad \sum_{u \in V} c_u s_u + \sum_{e \in E} c(e) x_e \tag{4–1}$$

$$\text{subject to } d_{uv} \leq s_u + s_v + x_{uv}, \qquad (u,v) \in E, \tag{4–2}$$

$$d_{uv} + d_{vw} \geq d_{uw}, \qquad (u,v) \in E, w \tag{4–3}$$

$$\sum_{u \neq v} d_{uv} \geq (1-\beta)\tbinom{n}{2}, \tag{4–4}$$

$$s_u \leq d_{uv} \leq x_{uv}, \qquad u, v \tag{4–5}$$

$$s_u, x_{uv} \in \{0,1\}, d_{uv} \in [0,1] \qquad u, v \tag{4–6}$$

where $s_u = 1$ if node u is removed and $s_u = 0$ otherwise. Similarly, $x_{uv} = 1$ indicates the removal of edge (u,v). The variables d_{uv} represent the disconnectivity (or distance) between nodes u and v in the residual network i.e. $d_{ij} = 1$ if i and j is disconnected and $d_{uv} = 0$ otherwise. The following lemma states the correctness of our formulation. The proof is similar to the case of the MILP for the β-vertex disruptor problem in [30], and is omitted here.

Lemma 8. *The optimal solution of ILP (5–8-5–9) induces a minimum cost β-disruptor $D_\beta = (V_\beta, E_\beta)$ of G, where $V_\beta = \{u \mid s_u = 1\}$ and $E_\beta = \{(u,v) \mid x_{uv} = 1\}$.*

4.1.2 Relation between edge costs and vertex costs

Since removing either u or v causes more disruption than removing the edge (u,v), we have the following lemma.

Lemma 9. *An edge $(u,v) \in E$ with $c(u,v) > \min\{c(u), c(v)\}$ will not appear in any optimal β-disruptor for any $\beta \geq 0$.*

The lemma reflects that vertices' costs are often higher than the costs of incident edges. Similarly, removing a vertex should not cost more than removing all the incident edges.

Lemma 10. *A vertex u with $c(u) > \sum_{(u,v)\in E} c(u,v)$ will not appear in any optimal β-disruptor for any $\beta \geq 0$.*

Lemmas 9 and 10 help us to exclude edges and vertices with "excessive" removal costs from further consideration. From the perspective of protecting the critical infrastructures, they provides relative caps for how much extra resource we should allocate to the network elements.

By definition, β-vertex disruptor can be seen as a special case of β-disruptor when all edges have infinity costs and β-edge disruptor is a special case of β-disruptor when all vertices have infinity costs. Since both vertex and edge disruptor are NP-hard, the β-disruptor problem is also NP-hard for $0 < \beta < 1$.

4.2 Bicriteria Approximation Algorithm for Joint Link and Node Attacks

In this subsection, we present an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. Since β-vertex disruptor is a special case of β-disruptor, the algorithm implies an $O(\sqrt{\log n})$ bicriteria approximation algorithm for β-vertex disruptor, which improve the best result for β-vertex disruptor, the $O(\log n \log\log n)$ bicriteria approximation algorithm in [33].

4.2.1 Algorithm Description

We will refer to the input network as the *original network*. We first reduce the β-disruptor problem in the original network to an instance of the β-edge disruptor problem in an *auxiliary directed graph*. The reduction maps each undirected edge to two *alternating directed edges* and each node to a *surrogate edge*. More importantly, we show that the reduction 'preserves' relative performance guarantees. We then apply a recursive cut procedure to find a near-optimal set of both alternating edges and surrogate edges that correspond to a β-disruptor in the original network.

Our algorithm JLNA(G) to find β-disruptor in directed graph G is summarized in Algorithm 8. In the first phase, the algorithm constructs an auxiliary graph $G\prime$ by splitting each vertex $v \in V$ into two new vertices v^+ and v^-. Formally, the set of vertices and edges in $G\prime$ are defined as

$$V\prime = \{\ v^-, v^+ \mid v \in V\}$$
$$E\prime = \{(v^-, v^+) \mid v \in V\} \cup \ \{(u^+, v^-) \mid (u, v) \in E\}$$

In addition, we assign costs $c\prime(.)$ for edges in $G\prime$: $c\prime(v^-, v^+) = c(v)$ for the surrogate edge (v^-, v^+) and $c\prime(u^+, v^-) = c(v^+, u^-) = c(u, v)$ for alternating edges (u^+, v^-) and (v^+, u^-). In the case, E is a mix of both undirected and directed edges, we also convert each directed edge $(p, q) \in E$ into an alternating edge $(p^+, q^-) \in E\prime$ with a cost $c\prime(p^+, q^-) = c(p, q)$.

In the second phase, the recursive cut procedure, shown in lines 4 to 11, construct a $\tilde{\beta}$-edge disruptor of $G\prime$, denoted by $E_{\tilde{\beta}}$. Here for a given $\beta\prime < \beta$, $\tilde{\beta} = \frac{1}{2}(\beta + \beta\prime)$. The $\tilde{\beta}$-edge disruptor is found by iteratively applying a subroutine SPARSE_CUT on the strongly connected components in $G\prime$. The subroutine SPARSE_CUT cut the components into smaller ones and the edges in a subset of the cuts are added to $E_{\tilde{\beta}}$. The process continues until the pairwise connectivity in the graph reduces to $\beta\binom{n}{2}$ or smaller. By the end of the second phase, $E_{\tilde{\beta}}$ is mapped back to edges and nodes in G to give a β-disruptor.

As shown in lines 4 and 5, the subroutine SPARSE_CUT is applied to each strongly connected component C to find a *minimum ratio cut* $\langle S\prime, \overline{S\prime}\rangle$ in C. The cut ratio for a cut is defined as follows.

Definition 6. *Let $G\prime = (V\prime, E\prime)$ be a directed graph. The ratio of a cut $\langle S\prime, \overline{S\prime}\rangle$ is $\alpha(S\prime) = \frac{c_{\text{out}}(S\prime)}{|S\prime||\overline{S\prime}|}$, where $c_{\text{out}}(S\prime)$ is the total cost of edges coming out from $S\prime$. In addition, a*

Algorithm 8: JLNA(G)

1. Construct the auxiliary graph $G\prime = (V\prime, E\prime)$
2. $\tilde{\beta} \leftarrow \frac{1}{2}(\beta + \beta\prime)$
3. $E_{\tilde{\beta}} \leftarrow \emptyset$
4. **for each** SCC C in $G\prime$
5. $\quad (C_E, C_\alpha) \leftarrow$ SPARSE_CUT(C)
6. **while** $\mathcal{P}(G\prime) > \tilde{\beta}\binom{n}{2}$
7. $\quad$ Find a SCC C^* of $G\prime$ with minimum cut ratio C^*_α
8. $\quad E_{\tilde{\beta}} \leftarrow E_{\tilde{\beta}} \cup C^*_E$
9. $\quad$ Remove edges in C^*_E from G
10. $\quad$ **for each** new component $C\prime$ in G
11. $\quad\quad (C\prime_E, C\prime_\alpha) \leftarrow$ SPARSE_CUT$(C\prime)$
12. $V_\beta \leftarrow \{v \mid (v^-, v^+) \in E_{\tilde{\beta}}\}$
13. $E_\beta \leftarrow \{(u, v) \mid (u^-, v^+) \in E_{\tilde{\beta}}\}$
14. **return** $D_\beta = (V_\beta, E_\beta)$

cut with the minimum cut ratio is called a minimum ratio cut *and denoted by*

$$\alpha(G\prime) = \min_{S\prime \subsetneq V\prime} \alpha(S\prime)$$

The output of SPARSE_CUT is a pair (C_E, C_α), where $C_E = \langle S\prime, \bar{S}\prime \rangle$ and $C_\alpha = \alpha(S\prime)$. For simplicity, we postpone the description of SPARSE_CUT til the proof on the approximation ratio.

In the main loop of JLNA, presented in lines 6 to 11, for each round we select, among the existing SCCs, a SCC C^* in G that has the smallest cut ratio. Let C^*_E and C^*_α be the cut set and the cut ratio of the cut found by SPARSE_CUT in C^*. We add C^*_E to $E_{\tilde{\beta}}$ and remove C^*_E from G. Removing $E_{\tilde{\beta}}$ breaks C^* into two or more strongly connected components. We again apply SPARSE_CUT on those components to find the minimum ratio cuts.

The main loop terminates when the pairwise connectivity in G is no more than $\beta\binom{n}{2}$. Then we construct the final solution by mapping each surrogate edge $(v^-, v^+) \in E_{\tilde{\beta}}$ to the node v in G, and each alternating edge $(u^-, v^+) \in E_{\tilde{\beta}}$ to the edge (u, v) in G.

4.2.2 Analysis of Approximation Ratio

We show that the JLNA algorithm is an $O(\sqrt{\log n})$ bicriteria approximation algorithm for the β-disruptor problem. We first show the connection between the cost of an optimal β-disruptor and the minimum cut ratio in Lemma 11. After that we derive the approximation ratio for JLNA in Theorem 4.1.

It is not obvious to see the connection between the cost of an optimal β-disruptor and the minimum cut ratio. Cuts in directed networks have different characteristics in comparison to their counterpart in undirected networks.

First, the cut ratios of $\langle S, \bar{S} \rangle$ and $\langle \bar{S}, S \rangle$ are different in general. In addition, different cuts may associate with the same set of links. For example, the cuts defined by $S =$ {blue nodes}, and $S =$ {blue and green nodes} associates to the same set of links $\{(u, v)\}$. To treat these differences, we use a randomized argument in the following lemma.

Second, components in directed networks are highly interdependent. As illustrated in Fig. 4-2, the failure of link (u, v) effectively breaks the network into four disconnected components. Red and green components loose the communication to other parts of the network, even none of their incoming and outgoing edges, colored in black, fail. In contrast, the only way to separate a component from the rest in undirected networks is to remove all links incident to the component.

Nevertheless, we are able to link the average cost to disrupt connected pairs in an optimal β-disruptor to the minimum cut ratio in the following lemma.

Lemma 11. *Given a directed graph $G = (V, E)$ and a subset of edges $M_\omega \subseteq E$, if $\omega = \mathcal{P}(G) - \mathcal{P}(G[E \setminus M_\omega]) > 0$, then $\frac{c(M_\omega)}{\omega} \geq \frac{1}{3}\, \alpha_{\min}(G)$, where*

$$\alpha_{\min}(G) = \min\{\alpha(C) \mid C \textit{ is a SCC of } G\}.$$

Proof. First, we prove the case G is strongly connected. When G is not strongly connected, the lemma can be proved by aggregating the results on SCCs of G.

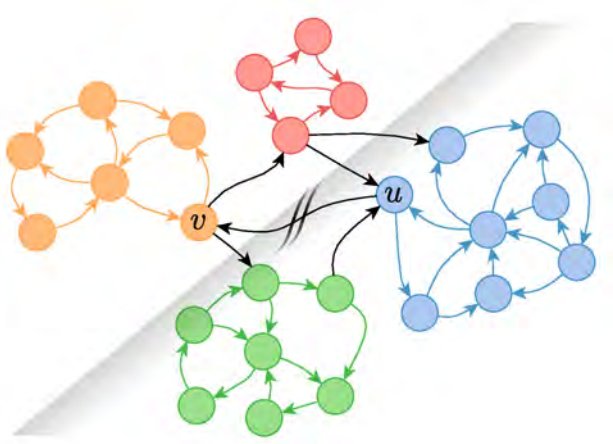


Figure 4-2. High interdependence of networks' elements. Removing the marked link (u, v) breaks the (strongly) connected network into four components. Notice that the red and green components are separated from the others, even when none of the incoming links to or outgoing links from those components are removed.

If G is strongly connected, then $\alpha_{\min}(G) = \alpha(G)$ and $\mathcal{P}(G) = \binom{n}{2}$. Let $C_1, C_2, \ldots, C_k$ be SCCs in $G[E \setminus M_\omega]$ and let $C_i(V)$ denote the set of vertices in component C_i. We have $\omega = \sum_{i<j} |C_i(V)||C_j(V)|$.

Observe that if we contract each SCC into a single node, we obtain the graph of SCCs which is a directed acyclic graph. Thus, there is a topological order for SCCs and we follow the convention that vertices with no incoming edges will have the smallest orders. Thus, w.l.o.g, we assume that the removed edges always come from SCCs with higher orders to SCCs with lower orders.

Consider all cuts $\langle S, \bar{S} \rangle$ of G that satisfy the follows

1Either $C_i(V) \subset S$ or $C_i(V) \subset \bar{S}$

2If $C_i(V) \subset S$ and there exists an edge from $C_i(V)$ to $C_j(V)$ in $G[E \setminus M_\omega]$, then $C_j \subset S$.

Clearly, $\langle S, \bar{S} \rangle \subseteq M_\omega$, hence, $c_{\text{out}}(S) \leq c(M_\omega)$. For a given pair of SCCs C_l and C_k, the probability that $C_l(V)$ and $C_k(V)$ belong to different sides of the cut is at least $1/3$. Since, there are four possible ways of assigning $C_l(V)$ and $C_k(V)$ to two sides of the cut, and at most one out of four is forbidden according to the second condition. Thus, $|C_l(V)||C_k(V)|$

pairs of vertices between C_l and C_k are separated with probability at least $1/3$. Hence, the expected number of pairs separated by a cut $\langle S, \bar{S} \rangle$ is at least

$$\mathbb{E}[|S||\bar{S}|] \geq 1/3 \sum_{i<j} |C_i(V)||C_j(V)| = 1/3\ \omega.$$

Among the cuts satisfied the two above conditions, there must be a cut $\langle S^*, \bar{S^*} \rangle$ that $|S^*||\bar{S^*}| \geq 1/3\ \omega$. Then,

$$\alpha(G) \leq \alpha(S^*) = \frac{c_{\text{out}}}{|S^*||\bar{S^*}|} \leq \frac{c(M_\omega)}{1/3\ \omega}$$

Hence, the lemma follows immediately.

Now, if G is not connected. Let $T_1, T_2, \ldots, T_l$ be SCCs of G, and let $M_\omega^{(j)}$ be the intersection of M_ω and the edges in T_j, and $T_{j\prime}$ be the subgraphs obtained from T_j after removing $M_\omega^{(j)}$. Apply the above result for the case the graph is connected on each connected component, we have

$$\begin{aligned} c(M_\omega) &= \sum_j c(M_\omega^{(j)}) \geq 3 \sum_j \alpha(T_j) \left(\mathcal{P}(T_j) - \mathcal{P}(T_{j\prime})\right) \\ &\geq 3\alpha_{\min}(G) \sum_j \left(\mathcal{P}(T_j) - \mathcal{P}(T_{j\prime})\right) \\ &= 3\alpha_{\min}(G) \left(\mathcal{P}(G) - \mathcal{P}(G[E \setminus M_\omega])\right) \end{aligned}$$

Thus, the lemma holds for every graph G. □

The quality and performance JLNA depend on the selection of SPARSE_CUT. For example, an exact algorithm to find minimum ratio cut will lead to a constant factor bicriteria approximation algorithm for β-disruptor. Unfortunately, finding the min ratio cut is an NP-hard problem [9]. Thus we have to rely on approximation algorithms to find good ratio cut in the graph.

Theorem 4.1. *For any fixed $0 \leq \beta\prime < \beta$, algorithm JLNA finds a β-disruptor of cost at most $O(\sqrt{\log n})\phi(\mathrm{OPT}_{\beta\prime})$, where $\mathrm{OPT}_{\beta\prime}$ is the cost of a minimum $\beta\prime$-disruptor.*

Proof. The proof consists of two steps. In the first step, we prove that $D_\beta = (V_\beta, E_\beta)$ is a β-disruptor of G. In the second step, we prove that the cost of D_β is at most $O(\sqrt{\log n})$ times the cost of a minimum $\beta\prime$-disruptor, denoted by $\mathrm{OPT}_{\beta\prime}$.

In order to prove that D_β is a β-disruptor of G, we show that the pairwise connectivity in G after removing edges in $G[-D_\beta] = (V \setminus V_\beta, E \setminus (E_\beta \cup V_\beta \times V_\beta))$ is at most $\beta\binom{n}{2}$. First, observe that vertices v^- and v^+ are either in the same SCC or they both are isolated. Here, we say a vertex is isolated if it belongs to a SCC of size one. Assume that $G\prime[E\prime \setminus E_{\tilde{\beta}}]$ can be decomposed into SCCs $C_1\prime, C_2\prime, \ldots, C_l\prime$ and $2t$ isolated vertices $w_1^-, w_1^+, \ldots, w_t^-, w_t^+$. Based on the construction of $G\prime$, we can verify that there are l corresponding SCCs $C_1, C_2, \ldots, C_l$ and t isolated vertices $w_1, w_2, \ldots, w_t$ in $G[-D_\beta]$. Moreover, $|C_i\prime| = 2|C_i|$ for $i = 1..l$.

Therefore, we have

$$\begin{aligned}\tilde{\beta}\tbinom{2n}{2} \geq \mathcal{P}(G\prime[E\prime \setminus E_{\tilde{\beta}}]) &= \sum_i \tbinom{|C_i\prime|}{2} \\ &= 4\sum_i \tbinom{|C_i|}{2} + \sum_i |C_i| = 4\mathcal{P}(G[-D_\beta]) + (n-t)\end{aligned}$$

Since $\tilde{\beta} < \beta$, we have

$$\mathcal{P}(G[-D_\beta]) \leq \frac{1}{4}\left(\tilde{\beta}\tbinom{2n}{2} - (n-t)\right) \leq \beta\tbinom{n}{2}$$

Thus, we have completed the first step. We prove the second step as follows.

Let $D^*_{\beta\prime} = (V_{\beta\prime}, E_{\beta\prime})$ be a minimum $\beta\prime$-disruptor i.e. $c(D^*_{\beta\prime}) = \mathrm{OPT}_{\beta\prime}$. Define

$$E\prime_{\beta\prime} = \{(v^-, v^+) \mid v \in V_{\beta\prime}\} \cup \{(u^+, v^-) \mid (u,v) \in E_{\beta\prime}\}.$$

By mapping SCCs of $G[-D^*_{\beta\prime}]$ to those of $G\prime[E\prime \setminus E\prime_{\beta\prime}]$ as in the first step, we can show that $E\prime_{\beta\prime}$ is a $\beta\prime$-edge disruptor of $G\prime$. Thus,

$$\mathrm{OPT}_{\beta\prime}(G) \leq \mathrm{OPT}^E_{\beta\prime}(G\prime).$$

Since $\beta\prime < \tilde{\beta}$, by Lemma 11 if removing a set of edges $M_\omega \subseteq E$ disrupts ω pairs of vertices, then $\frac{c(M_\omega)}{\omega} \geq 1/3\alpha_{\min}(G)$. At any round in the while loop of RBA, since a set of edges $E^*_{\beta\prime}$ in a minimum $\beta\prime$-edge disruptor, for some $0 < \beta\prime < \beta$, can disrupt at least $(\beta - \beta\prime)\binom{n}{2}$ more pairs in G, we have

$$\mathrm{OPT}^E_{\beta\prime} / \left((\beta - \beta\prime)\tbinom{n}{2}\right) \geq 1/3\alpha_{\min}(G). \tag{4–7}$$

Since our cut procedure is an $O(\sqrt{\log n})$ factor approximation algorithm for the min cut ratio problem, the average cost to disrupt a pair by removing C^*_E is upper bounded by $O(\sqrt{\log n})\alpha_{\min}(G)$. By (4–7), the average cost to disrupt pairs in the graph at any step is at most $O(\sqrt{\log n})(\mathrm{OPT}^E_{\beta\prime}) / \left((\beta - \beta\prime)\binom{n}{2}\right)$. Therefore, even when E_β disrupt all $\binom{n}{2}$ pairs in G, the total cost is no more than

$$O(\sqrt{\log n}) \times \frac{\mathrm{OPT}^E_{\beta\prime}}{\left((\beta - \beta\prime)\binom{n}{2}\right) \times \binom{n}{2}} \leq \frac{O(\sqrt{\log n})}{(\beta - \beta\prime)} \times \mathrm{OPT}^E_{\beta\prime}.$$

Thus we have

$$c(E_{\tilde{\beta}}) \leq O(\sqrt{\log 2n})\mathrm{OPT}^E_{\beta\prime}(G\prime) \leq O(\sqrt{\log n})\mathrm{OPT}_{\beta\prime}(G).$$

That yields the proof. □

4.3 Hybrid Meta-heuristic

Our second choice for SPARSE_CUT is a simple yet efficient spectral bisection method [61]. The β-edge disruptor found by RBA is further optimized by a hybrid of variable neighborhood search [60] and simulated annealing [51]. Numerical results in Section 6.4 suggest that our hybrid method is competitive for the β-edge disruptor problem.

4.3.1 Spectral Bisection

Let $A = \{c_{ij}\}$ be the cost matrix of $G = (V, E)$ where $c_{ij} = c(v_i, v_j)$ is the cost of edge (v_i, v_j) and $c_{ij} = 0$ if $(v_i, v_j) \notin E$. The unnormalized graph Laplacian matrix [61]

Algorithm 9: Spectral_Bisection(G)

Compute eigenvector x corresponding to λ_2 of L
Sort entries in x
for $p = 1$ to n
    Calculate ratio of the cut $S_p = \{v_i \mid x_i \leq x_p\}$
return S_p with the best ratio cut

is defined as $L = D - A$, where D is a diagonal matrix with the weighted degrees of vertices on the diagonal.

The matrix L is symmetric and positive semi-definite, since for every vector $x \in \mathbb{R}^n$ we have

$$x^T L x = \frac{1}{2} \sum_{i,j=1}^{n} c_{ij}(x_i - x_j)^2 \geq 0. \tag{4–8}$$

L has n non-negative, real-valued eigenvalues $\lambda_1 = 0 \leq \lambda_2 \leq \ldots \leq \lambda_n$. The second smallest eigenvector of L, λ_2, is known as the algebraic connectivity of the graph and can be used to describe many properties of graphs [61]. We shall use the eigenvector corresponding to λ_2 to derive the bisection of vertices in G.

Recall that SPARSE_CUT aims to find the min ratio cut

$$\min_{S \subsetneq V} \frac{c(S, \bar{S})}{|S||\bar{S}|} \tag{4–9}$$

Consider a vector $x \in \{0, 1\}^n$ represent a set of vertices in S i.e. $x_i = 1$ if $v_i \in S$ and $x_i = 0$ otherwise. We rewrite the min ratio cut problem as

$$\min_{x \in \{0,1\}^n, x \neq 0, \mathbb{1}} \frac{\sum_{(v_i, v_j) \in E} c_{ij}(x_i - x_j)^2}{\sum_i \sum_j (x_i - x_j)^2} \tag{4–10}$$

Since the problem is NP-hard, we relax the condition $x_i \in \{0, 1\}$ to $x_i \in [0, 1]$. Substitute x with vector $y = x - \frac{\|x\|_1}{n}$. After some algebra, we obtain an equivalent problem of (4–10)

$$\min_{y \neq 0, y \perp \mathbb{1}} \frac{1}{n} \frac{y^T L y}{y^T y} \tag{4–11}$$

By Courant-Fisher theorem [61], the solution of the above minimizing problem is exactly the eigenvector corresponding to the second smallest eigenvalue of λ_2. So we can approximate the optimal solution of the min ratio cut problem with the second eigenvector of L by transforming the real-valued x into a zero-one vector. One simple way is to sort the x_i to give a linear ordering of the vertices then determine the splitting index p that yields the best cut ratio. The whole procedure is summarized in Algorithm 9.

Assume that the eigenvalues can be found within a constant number of iterations [54], RBA algorithm will have an $O(n^2)$ time complexity.

4.3.2 Hybrid Meta-heuristic

As we cannot control how many connected pairs SPARSE_CUT will separate, RBA algorithm usually disrupts more connected pairs than required, resulting in less optimal solutions. Therefore, in order to further improve the performance of RBA, we introduce a hybrid method, using both simulated annealing [51] and variable neighbourhood search (VNS)[60]. The simulated annealing makes the number connected pairs converge to the desired level, while the local search methods explore alternative solutions to reduce the cost.

For β-edge disruptor problem, multiple neighborhood structure is essential to obtain high quality solutions. To find minimum β-edge disruptor, we aim to minimize the cut ratio that may lead to disrupting more pairs than necessary and incurring higher costs. Alternating among neighborhood structures enables us to seek for edge disruptors with both small ratio cuts and small costs. Similar to simulated annealing, we allow “uphill” moves that increases the cost of the solution if they improve certain aspects of the solution.

We consider four different neighborhood structures. From a solution or a partial solution $E_\beta \subset E$, the set of neighbors in each neighborhood structure can be obtained as follows

Algorithm 10: HMH(G)

$E_\beta \leftarrow \emptyset, \tau \leftarrow \min\{\beta, 1-\beta\}$
while $\tau > 1/\binom{n}{2}$
  $\tau \leftarrow \frac{1}{2}\tau$
  $E_\beta \leftarrow E_\beta \cup$ RBA$(G[E \setminus E_\beta], \beta - \tau)$
  for $k = 1$ to 3 /* Phase 1: Condensation */
    repeat
      Consider all type k neighbors $E\prime_\beta$ that
      $c(E\prime_\beta) \leq c(E_\beta)$ and $\mathcal{P}(G[E \setminus E\prime_\beta]) \leq \beta\binom{n}{2}$
      Find among them $E\prime_\beta$ with the smallest cut ratio
      $E_\beta \leftarrow E\prime_\beta$
    until no change in E_β
  for $k = 1$ to 4 /* Phase 2: Exploration */
    repeat
      Consider all type k neighbors $E\prime_\beta$ that
      $(\beta - \tau)\binom{n}{2} \leq \mathcal{P}(G[E \setminus E\prime_\beta]) \leq (\beta + \tau)\binom{n}{2}$
      Find among them $E\prime_\beta$ with the smallest cut ratio
      $E_\beta \leftarrow E\prime_\beta$
    until no change in E_β
return the best solution so far

- Type 1: Merge two connected components in $G[E \setminus E_\beta]$ i.e. remove the edges between them from E_β.
- Type 2: Move a vertex from one component to an adjacent component in $G[E \setminus E_\beta]$.
- Type 3: Swap places of two adjacent vertices (u, v) which belong to two different components.
- Type 4: Partition a component in $G[E \setminus E_\beta]$ with Spectral_bisection.

Beside reducing the total cost, we also want to move to neighbors with smaller cut ratio which is defined as

$$\alpha(E_\beta) = \frac{c(E_\beta)}{\mathcal{P}(G) - \mathcal{P}(G[E \setminus E_\beta])}$$

The cut ratio is the average cost to disrupt pairs by removing edges in E_β. We use the *best improvement* strategy i.e. among eligible neighbors we change to the neighbor with the smallest cut ratio.

Our hybrid meta-heuristic (HMH) is presented in Algorithm 10. We use a parameter τ, similar to the *heating condition* in Simulated Annealing [51], to control how far the pairwise connectivity in the graph can diverge from the target connectivity $\beta\binom{n}{2}$. Every round, τ is reduced by half until it is negligibly small. The algorithm alternates between two phases: *condensation* and *exploration*. In the condensation phase, the goal is to reduce the cost of the current β-edge disruptor. As mention, we do not favor the neighbor with the largest decrease in cost but the one with the smallest cut ratio.

In the exploration phase, we emphasize on improving the cut ratio to find potential good partition of the network. Moving to neighbors with higher costs is possible during this phase as long as the pairwise connectivity differs at most $\tau\binom{n}{2}$ from the target connectivity level $\beta\binom{n}{2}$. If outcome of the exploration phase is not a β-edge disruptor, the RBA algorithm is invoked to produce a greedy solution before the algorithm continues the condensation phase again. Finally, the algorithm output the smallest cost β-edge disruptor, encountered during the search.

Since the algorithm has at most $\log\binom{n}{2} = O(\log n)$ phases, and it spends at most $O(n^3)$ times to improve the solution within each phase, the HMH algorithm has a time complexity $O(n^3 \log n)$. In our experiments, it has (almost) the same running time with RBA using spectral bisection in place of HMH, which has an $O(n^2)$ time complexity.

Directed HMH Algorithm. The algorithm to find a sparse cut in [2] has a high time complexity $O(n^{9.5})$ as it requires solving a large Semidefinite Programming. Fortunately, we again can turn to spectral partitioning to find small ratio cuts in directed graphs and further optimize the solution using the similar techniques in Algorithm HMH.

The major change is to replace the spectral bisection in undirected graph with one for directed graph. This can be done by transforming the asymmetric adjacency matrix A to a symmetric one using one of the symmetrization methods such as $(A + A^T)/2, AA^T$, etc. [57]. Besides, we consider a new type of neighborhood (Type 5), in which we can

remove (or un-remove) a node in the graph. With one of the mentioned symmetrization methods, the HMH algorithm for directed graph also has the time complexity $O(n^3 \log n)$

4.4 Experimental Studies

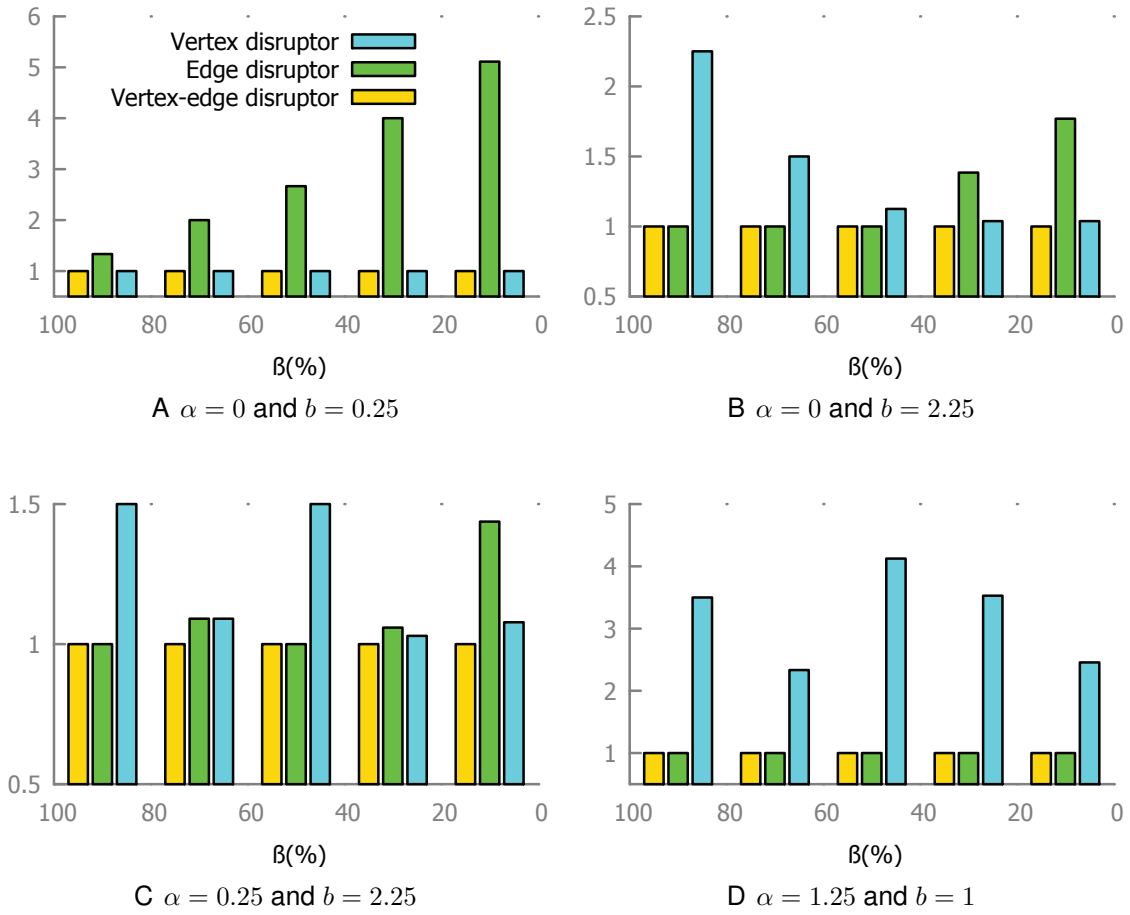


Figure 4-3. The normalized optimal costs of three different disruptor types on the US Backbone network.

We illustrate through our experiments the need to assess network vulnerability under joint node and link attacks.

4.4.1 Experiment Setups

4.4.1.1 Datasets

The experiments are performed on three real communication networks, namely IP Backbone[1], CAIDA AS[56], and Oregon AS[56], and a set of four synthesis

networks described in subsection 4.4.3. The network details are given in the subsequent subsections and the references.

4.4.1.2 Removal costs schemes

Assigning meaningful costs for edges and vertices is a challenging task which usually depends on the availability of the data. For simplicity, we assume that all edges has uniform removal costs $c(e) = 1 \forall e \in E$. Note that we can always multiply simultaneously edge and vertex costs with a constant, then all optimal disruptors stay optimal (with the costs multiplied by the same constant). We assign the cost of removing a vertex u to be $c(u) = b + \alpha d(u)$, where b and α are non-negative constants. In other words, attacking a node requires paying a base cost b and an extra cost that is proportional to the degree centrality. Other centrality measurements e.g. PageRank, Betweeness centrality can also be used in place of $d(u)$ to weight the u's importance.

4.4.1.3 Finding the optimal disruptor

The optimal solutions are found by solving the MILP in Section 4.1.1 with the sparse metric and advanced plane cutting techniques in our previous work [30]. The mathematical optimization package to solve the IP is GUROBI 4.5.

4.4.1.4 Solving for the second eigenvector

The major time of HMH (Algorithm 10) spends on finding the second smallest eigenvector of the Laplacian matrix. The eigenvectors are found using the *Implicitly Restarted Arnoldi Method*, implemented in ARPACK [54]. We use SuperLU [29] as the linear systems solver.

We use the Shift and Invert spectral transformation to enhance the convergence rate [1] . We select a scalar $\sigma = 0.01$, called the *shift*, and transform the original problem $Lx = \lambda x$ into the shift-and-invert problem $(L - \sigma I)^{-1}x = \mu x$ where $\mu = 1/(\lambda - \sigma)$. Note

[1] In many cases, the regular mode does not converge after 20,000 iterations.

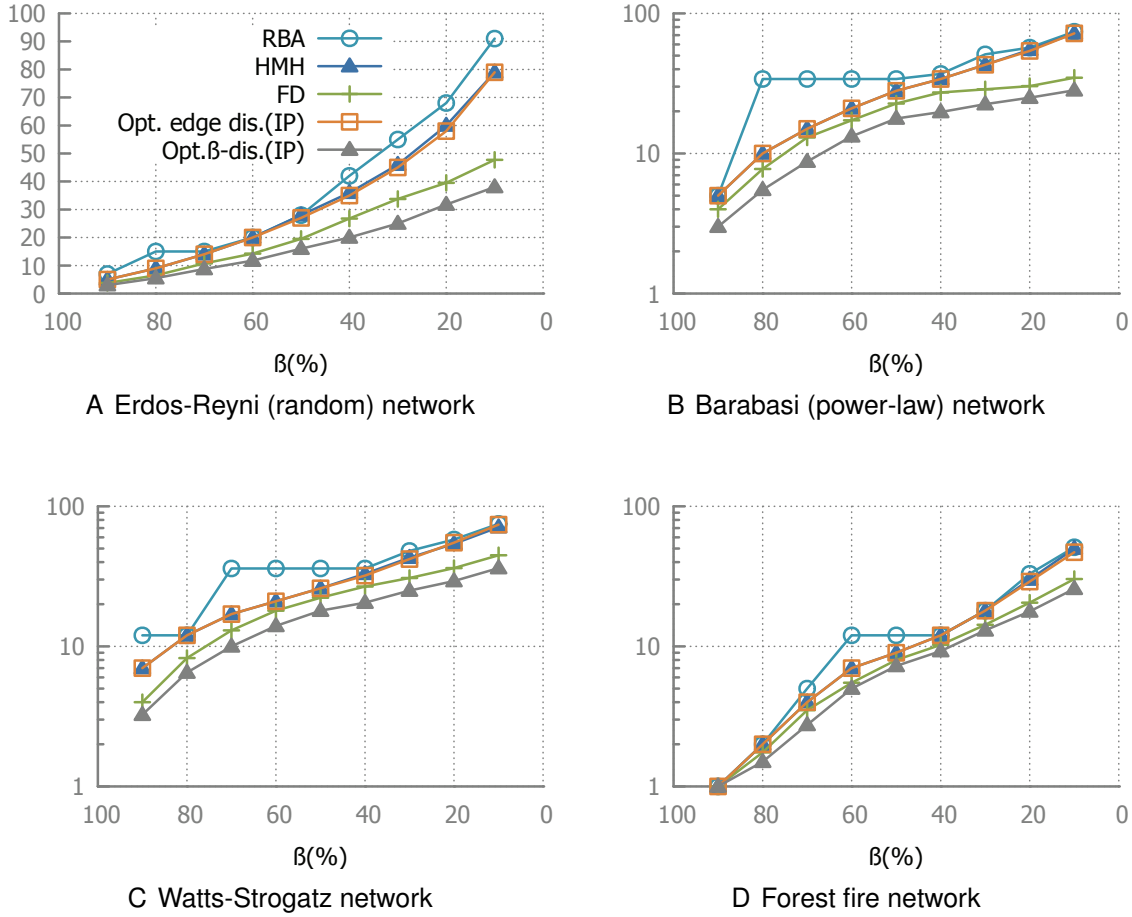


Figure 4-4. Costs of disruptor algorithms on the synthesis networks

that setting $\sigma = 0$ will crash ARPACK since L is non-invertible (sum of rows equal zero) [2] .

In the case of JLNA, spectral bisection is performed on the symmetrized matrix $A\prime + A\prime^T$, where $A\prime$ is the adjacency matrix of the auxiliary graph $G\prime$, constructed in Algorithm 8.

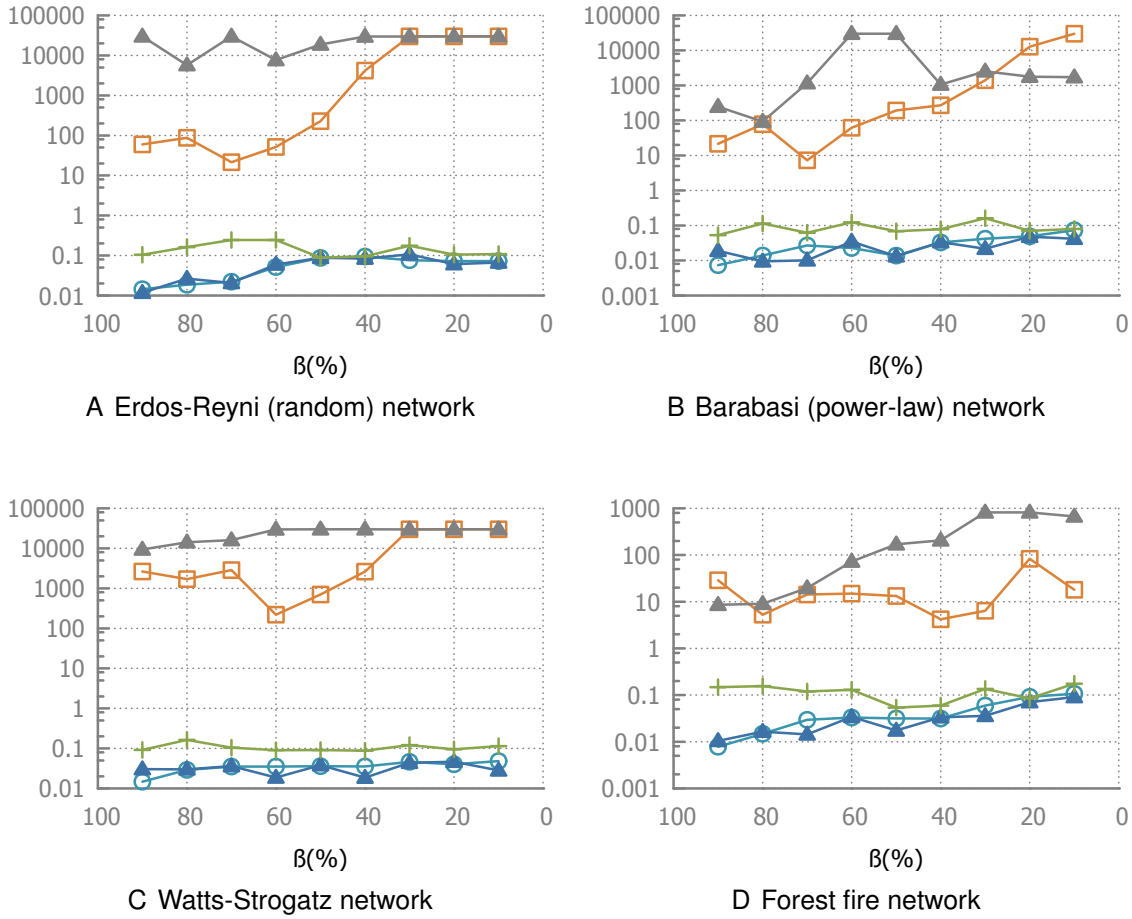


Figure 4-5. Running time of disruptor algorithms on the synthesis networks

4.4.1.5 Implementation details

All algorithms are implemented in C++ and compiled with GCC 4.4 compiler on a 64 bit Linux machine with a Quad-core AMD Opteron 2350 2.0 Ghz processor and 32 GB memory. Only a single core is used during the experiments.

4.4.2 Comparison of the three disruptor types

Before analyzing the experimental results, given in Fig. 4-3, for three different disruptor types (edge, vertex, and general), we summarize the provable connections

[2] The 'eigs' function to find eigenvalues in MATLAB crashes for this reason.

among those three. First, the cost of optimal β-disruptor is always less than the costs of both β-edge disruptor and β-disruptor. Second, if the cost scheme $c(u) = b + \alpha d(u)$ is in use, we can tell when the cost of the optimal $beta$-disruptor meets exactly the minimum of that of β-edge disruptor and β-vertex disruptor.

Note that if $c(u) < c(u, v)$ for some $(u, v) \in E$, then the edge (u, v) should not be removed (as we can remove u instead). Similarly, a node u with $c(u) > \sum_{(u,v)\in E} c(u, v)$ will not be removed since we can always remove all of its incident edges. Therefore, we obtain the following properties.

- $\alpha = 0,\ b \leq 1$: $\mathrm{OPT}_\beta = \mathrm{OPT}_\beta^V \leq \mathrm{OPT}_\beta^E$ i.e. the optimal β-disruptor contains *no edges*.
- $\alpha = 0,\ b > 1$: the optimal solutions contain no u with $d(u) < b$.
- $0 < \alpha < 1$: the optimal β-disruptor contains only vertices of degree at least $\frac{b}{1-\alpha}$.
- $1 \leq \alpha$: $\mathrm{OPT}_\beta = \mathrm{OPT}_\beta^E \leq \mathrm{OPT}_\beta^V$ i.e. the optimal β-disruptor contains *no vertices*.

We test four different settings of α and b that correspond to the above four cases on the fiber backbone operated by a major U.S. network provider [1]. The optimal costs of three disruptor types are shown in Fig. 4-3. In Fig. 4-3A, the costs of β-disruptor equal exactly the cost of β-vertex disruptor; in Fig. 4-3D, the costs of β-disruptor equal exactly the cost of β-edge disruptor. These agree with the above four mentioned cases. However, for Figs. 4-3B and 4-3C, the costs of β-disruptor are *strictly less than* the minimum of both edge-disruptor and vertex-disruptor. In addition, for small β the cost of edge-disruptor is less than that of vertex-disruptor, while for large β the vertex-disruptor has substantially smaller cost. This suggests that small scale attacks should target links, while large scale attacks should pay more attention to nodes to reduce the attack cost. Nevertheless, a combination of both node and link attacks would result in a more cost-effective strategy to break the network.

4.4.3 Synthesis Networks of Different Topologies

We test our algorithms on *synthesis moderate-sized* networks to 1) compare the solutions of our algorithms to that of the optimal solutions obtained by solving Integer programming (IP), and 2) verify the performance of the algorithms across different network topologies. Four synthesis networks of 100 nodes and approximately 200 edges are generated following below complex network models.

- **Erdos-Reyni**: A random graph of 100 vertices and 200 edges following the Erdos-Reyni model [36].
- **Barabasi-Albert**: A power-law model using preferential attachment mechanism [12].
- **Watts–Strogatz**: A random graph which exhibit small-world phenomenon following model [79] with the dimension of the lattice 2 and the rewiring probability 0.3[79].
- **Forest fire**: A random power-law graph following Forest fire model by Leskovec et al. [56] with the forward and backward burning probabilities 0.3 and 0.9, respectively.

Set up. We show the costs produced by both types of disruptors β-edge disruptor and β-disruptor in Fig. 4-4. The measured β-edge disruptor algorithms are RBA (the RBA algorithm using spectral bisection in the place of SPARSE_CUT), HMH, and optimal β-edge disruptor (Opt. edge dis.) ; and the measured β-*disruptor* algorithms are JLNA and optimal β-disruptor (Opt.β-dis.). The costs of vertices follow the linear scale $c(u) = b + \alpha d(u)$ where $\alpha = 0.25$ and $b = 0.25$.

β-*edge disruptor.* Among β-edge disruptor algorithms, HMH matches the optimal solutions obtained by solving IP (Opt. edge dis) most of the times; and in other cases, the gap between the two are negligibly small. However, there are much larger gaps between RBA's solutions and the optimal solutions. The reason RBA is less competitive is that it may separate many more node pairs than the required, especially when β is small. For example, when $\beta = 80\%$ the cost of RBA is more than three times higher than that of the HMH algorithms and the optimal solutions. This implies that the annealing

procedure and the VNS in HMH are capable of correcting the overcutting caused by RBA.

β-disruptor. The β-disruptor costs found by JLNA also closely approach those of the optimal (Opt. β dis.). On average, the solution of JLNA is only 15% larger than the optimal. However, the gap between JLNA and the optimal is less impressive than that between HMH and the optimal edge disruptor. The reason is possibly due to the fact that we simply symmetrize the adjacency matrix to find the directed cut; and the JLNA's performance can be enhanced with better directed spectral cut algorithms. Nevertheless, the cost of JLNA is substantially less than that of the optimal edge disrutpor. Thus, JLNA is able to reveal vulnerability at both links and nodes in the network.

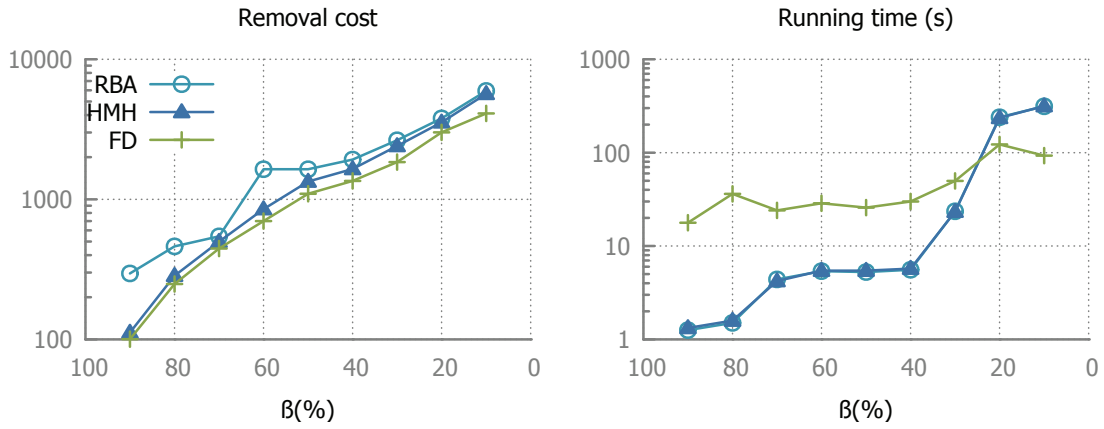


Figure 4-6. Oregon AS network

Running time. Fig. 4-5 shows the running time on four synthesis networks. Both the two IPs take excessive amounts of time (up to 10 hours) to return the solution on Erdos-Reyni, Barabasi and Watts-Strogatz network. In contrast, RBA, HMH and JLNA take less than one second to complete in all cases. Since JLNA algorithm splits the nodes in the network (thus, double the network size), it has slightly higher running time than those of RBA and HMH.

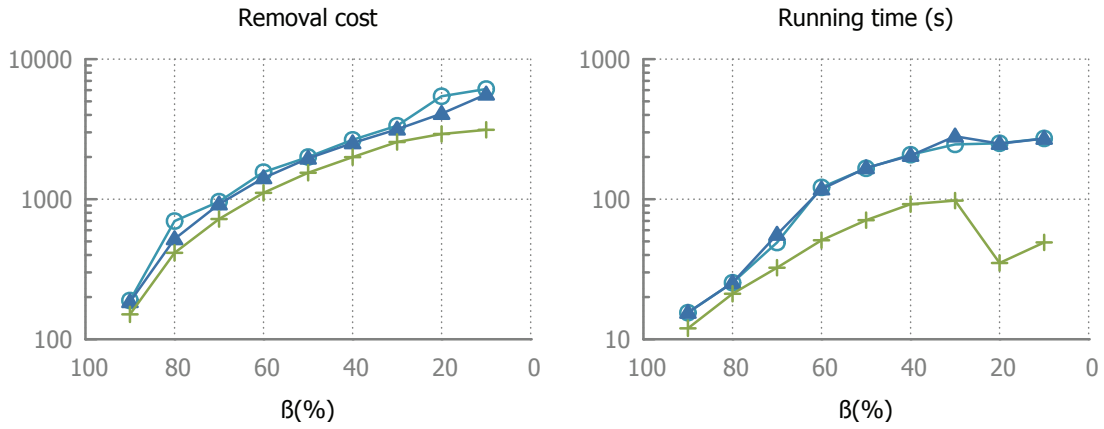


Figure 4-7. CAIDA AS network

Overall, HMH and JLNA algorithms prove to be excellent choices to find disruptors in moderate-sized networks. They produce high quality solutions within a short amount of time and the performance is stable across different network topologies. We further study the performance of HMH and JLNA on larger (real) communication networks.

4.4.4 AS Relationships Networks

We analyze HMM, and JLNA [3] , with the same settings in the last subsection, on the following AS relationships datasets.

- **CAIDA AS**: The CAIDA AS Relationships Dataset from Sep. 17, 2007 [56] with 8,020 nodes and 36,406 links.
- **Oregon AS**: AS peering information inferred from Oregon route-views between March 31 and May 26, 2001 [56]. Only the largest connected component with 11,174 nodes and 23,410 links is considered.

The costs and running times are reported in Figs. 4-6 and 4-7. Similar to the cases of synthesis networks, the HMH algorithm continue to produces solutions with significantly smaller costs than RBA's. For $\beta = 80\%$ and $\beta = 60\%$, the cost of HMH is less than half of that in RBA. In addition, both algorithms have almost identical running time. The major portion of time is spent on finding the second eigenvector; and performing local search

[3] The IPs cannot handle networks with more than few hundred nodes.

and annealing procedure is relatively inexpensive. Obviously, HMH dominates RBA with higher quality solutions and within (almost) the same running time.

Note that computing eigenvector, the bottleneck in our algorithms, can be done efficiently in a distributed manner. For example, Google is capable of solving the eigenvalue problem on the web network of billions of nodes. Thus, the proposed solutions are scalable for much larger networks.

Joint node and link attacks pose a serious threat to the network. In addition to network connectivity, it is also important to assess the vulnerability of the network under joint node and links networks in terms of other performance metrics such as network throughput, maximum network flow between source-destination pairs, and so on. Furthermore, the problem of allocating resource to protect the network under the joint attacks is of great importance and is the topic of our future study.

CHAPTER 5
VULNERABILITY ASSESSMENT IN PROBABILISTIC NETWORKS

We investigate the vulnerability of *probabilistic networks* under multiple attacks. That is we aim to identify the most critical subsets of infrastructure whose removal maximize the disruptive effect on the network in term of connectivity. Finding such a subset is extremely challenging due to the uncertainty of the network topology and the exponentially large number of attack schemes. We show that finding exact solution is computationally intractable and propose an efficient *two-stage stochastic programming* to approximate the identification of the critical infrastructure. Furthermore, we propose a novel *sampling scheme*, which find solutions with guaranteed probabilistic accuracy. Finally we demonstrate the effectiveness and efficiency of the proposed algorithms on real and synthetic data sets.

Disruptive events, ranging from natural disasters to malicious attacks, can drastically compromise the network's ability to meet its quality-of-service(QoS) requirements, if not cause widespread service outages and potentially total network breakdown [46, 62, 64, 68]. Moreover, there is a significant concern over critical infrastructures in electrical power grids and highway systems as targets for terrorist attacks [67]. To mitigate the risk and develop proactive responses, it is essential to assess network vulnerability to identify the most destructive attack scenarios.

Although there has been a significant amount of work on assessing network vulnerability, most previous works focus mainly on using centrality measurements e.g. degree, betweeness, and closeness centralities [6, 7] to identify critical links or nodes. Unfortunately, these approaches only determine the relative importance of *a small number* of nodes or links and cannot reveal the enormous damage potential caused under *simultaneous* attacks. Other set of works studies links and nodes removal problems that optimize several global graph measures, such as clustering coefficient, network diameter, etc. However, these measures do not cast well for particular kinds

of network vulnerability, when the network connectivity is of high priority. To this end, *pairwise connectivity*, the number of node pairs that remain connected, has been recently used as an effective measure to account for the effect of the attacks [11, 30, 33, 62, 64].

5.1 Probablilistic Networks

In this section, we first present the considered probabilistic network model, followed by the formal definition of the studied problems.

5.1.1 Probabilistic Network Model

The network with uncertain links is modeled using a tuple $\mathcal{G} = (V, E, p)$ where vertices in V corresponds to the set of nodes, edges in E corresponds to the set of links in the network, and $p : E \to [0, 1]$ maps each edge $(u, v) \in E$ to a real number in $p_{uv} \in [0, 1]$ that represents the availability of (u, v) and $p_{uv} = 0$ for all $(u, v) \notin E$. Further, denote by ξ the adjacency matrix of $\mathcal{G}$, i.e., $\mathsf{Pr}[\xi_{uv} = 1] = p_{uv}$ and $\mathsf{Pr}[\xi_{uv} = 0] = 1 - p_{uv}$ for all pairs (u, v). For clarity, we consider only undirected networks and assume that the existing of edges are independent of one another, though our approaches also apply in principle to directed graphs or graphs with edge correlations as long as we can effectively generate *samples* of the probabilistic graph.

A *sample graph* (or a realization) $G^l = (V, E^i)$ of $\mathcal{G}$ is generated by selecting each edge $e \in E$ with probability $p(e)$. The sample space $\mathcal{S}_\mathcal{G}$ consists of $N = 2^{|E|}$ possible samples $\mathcal{S}_\mathcal{G} = \{G^1 = (V, E^1), G^2 = (V, E^2), \ldots, G^N = (V, E^N)\}$ of G that correspond to $2^{|E|}$ possible subsets of E. The probability that G^l is sampled from $\mathcal{G}$ is given by

$$f_\mathcal{G}(G^l) = \mathsf{Pr}[G = G^l] = \prod_{e \in E^i} p_e \prod_{e \in E \setminus E^i} (1 - p_e)$$

Moreover, the matrix $\{\xi^l\}_{uv}$ are used to denote the adjacency matrix of G^l.

5.1.2 Expected Pairwise Connectivity

As mentioned above, our measure for the disruptive effect is based on the value of *pairwise connectivity* (EPC), which is the number of (expected) connected pairs in the

residual network. For a deterministic graph G^l, the *pairwise connectivity*, denoted by $\mathcal{P}(G^l)$, is the number of pairs (u, v) with at least one path between u and v. Naturally, the *expected pairwise connectivity* for the probabilistic graph $\mathcal{G}$ is defined as

$$\mathsf{EPC}(\mathcal{G}) = \mathbb{E}\left[\mathcal{P}(\mathcal{G})\right] = \sum_{l=1}^{N} f_{\mathcal{G}}(G^l)\mathcal{P}(G_i).$$

Lemma 12. *Given a probabilistic graph $\mathcal{G} = (V, E, p)$, we have*

$$\mathsf{EPC}(\mathcal{G}) = \frac{1}{2} \sum_{u,v \in V; u \neq v} \mathsf{REL}_{u,v}(\mathcal{G})$$

where $\mathsf{REL}_{u,v}(\mathcal{G})$ is the probability that v is reachable from u within $\mathcal{G}$.

Partial order. Given two probabilistic graphs $\mathcal{G}_A(V_A, E_A, p^A)$ and $\mathcal{G}_B(V_B, E_B, p^B)$, we say $\mathcal{G}_A$ is *dominated* by $\mathcal{G}_B$, and write $G_A \preceq G_B$, iff $V_A \subseteq V_B, E_A \subseteq E_B$, and $p_e^A \leq p_e^B \ \forall e \in E_A$.

Lemma 13. *Given two probabilistic graphs G_A and G_B, if $G_A \preceq G_B$, then* $\mathsf{EPC}(\mathcal{G}_A) \leq \mathsf{EPC}(\mathcal{G}_B)$.

5.1.3 Vulnerability Assessment

We define the following problems based on their deterministic versions in [10, 33]. On one hand, the number of nodes/ edges to remove is given, and we wish to maximize the *expected* disruptive effect.

k-Probabilistic Critical Nodes Problem (k-pCNP). Given a probabilistic network $\mathcal{G} = (V, E, p)$ and an integer $0 \leq k \leq n$, find a k nodes subset $S \subset V$ that removal minimizes the expected pairwise connectivity in the residual network after removing the nodes in S.

5.2 Estimation of Connectivity in Probabilistic Networks

5.2.1 #P-Completeness

In this paper, we first show that computing expected pairwise connectivity in a probabilistic network is #P-complete. A computation problem f in #P is said to be #P-complete if every problems in #P is reducible to f. Here #P is the class of "counting

version" of problems in NP, i.e., they are problems of the form "compute the number of solutions" for a problem in NP. Showing that a computation problem is #P-complete makes a strong statement about its intractability: if such a problem were computable in polynomial time then not only P=NP but also P=PH.

Theorem 5.1. *Computing the expected pairwise connectivity* $\mathrm{EPC}(G)$*, given a probabilistic graph* G*, is #P-complete.*

Proof. We prove the theorem by a reduction **from** the counting problem of $s-t$ connectedness in an undirected graph [75]. The problem is to count the number of subgraphs of a graph G in which there is a path from s to t. The problem is equivalent to computing the probability that s is connected to t when each edge in G has an independent probability $1/2$ of being connected, and another $1/2$ to be disconnected.

We reduce this problem to the expected pairwise connectivity computation problem as follows. We first construct four probabilistic graphs G_0, G_1, G_2, G_3, where

- $G_0 = G$ and $p(e) = 1/2$ for all $e \in E$.
- G_1 is obtained from G_0 by adding a new node $s\prime$ and an edge $(s, s\prime)$ with $p(s, s\prime) = 1$.
- G_2 is obtained from G_0 by adding a new node $t\prime$ and an edge $(t, t\prime)$ with $p(t, t\prime) = 1$.
- G_3 is obtained from G_0 by adding nodes $s\prime$ and $t\prime$, and edges $(s, s\prime)$, and $(t, t\prime)$ with probabilities $p(s, s\prime) = p(t, t\prime) = 1$.

Next we compute $P_0 = \mathrm{EPC}(G_0), P_1 = \mathrm{EPC}(G_1), P_2 = \mathrm{EPC}(G_2)$, and $P_3 = \mathrm{EPC}(P_3)$. Then, we can return $P_0 - P_1 - P_2 + P_3$ as the probability that s is connected to t and thus we solve the $s-t$ connectedness counting problem. In addition, there is an obvious reduction from the expected pairwise connectivity computation problem **to** the $s-t$ connectedness problem via the equality $\mathrm{EPC}(\mathcal{G}) = \sum_{u \neq v} \mathrm{REL}_{u,v}(\mathcal{G})$. It is shown in [75] that $s-t$ connectedness is #P-complete, and thus the expected pairwise connectivity computation problem is also #P-complete.

Finally, we prove that $P_0 - P_1 - P_2 + P_3 = \mathsf{REL}_{s,t}(G)$, the probability that s and t is connected in G. By the construction of G_1, we have $P_1 = P_0 + \sum_{v \in V} \mathsf{REL}_{st,v}(G) = P_0 + \sum_{v \in V} \mathsf{REL}_{s,v}(G)$. Similarly, $P_2 = P_0 + \sum_{v \in V} \mathsf{REL}_{v,t}(G)$ and $P_3 = P_0 + \sum_{v \in V} \mathsf{REL}_{s,v}(G) + \sum_{v \in V} \mathsf{REL}_{v,t}(G) + \mathsf{REL}_{s,t}(G)$. It is straightforward to verify that $\mathsf{REL}_{s,t}(G) = P_0 - P_1 - P_2 + P_3$. □

We are interested in (ϵ, δ)-approximations for $\mathsf{EPC}(\mathcal{G})$, i.e., algorithms returning an estimate of $\mathsf{EPC}(\mathcal{G})$ accurate to within a relative error of ϵ with probability at least $1 - \delta$. Formally, we define (ϵ, δ)-approximations as follows.

Definition 7 ((ϵ, δ)-approximation)**.** *A function $\hat{F}(G)$ is an (ϵ, δ)-approximation for the expected pairwise connectivity $\mathbb{E}[\mathcal{P}(G)]$ if*

$$\textit{Pr}\left[(1-\epsilon)\mathbb{E}[\mathcal{P}(G)] \leq \hat{F}(G) \leq (1+\epsilon)\mathbb{E}[\mathcal{P}(G)]\right] > 1 - \delta$$

An (ϵ, δ)-approximation is called a *fully polynomial randomized approximation scheme* (FPRAS) if its running time is bounded by a polynomial in $1/\epsilon, log(1/\delta)$, and the input size. An FPRAS is generally regarded as a robust notion of "approximation algorithm" for counting problems. Sinclair and Jerrum showed that every #P-complete problem either has an FPRAS, or is essentially impossible to approximate [70].

5.2.2 Monte-Carlo Methods to Approximate EPC

We present a simple Monte-Carlo algorithm to estimate the EPC in Algorithm 11. The algorithm draws $N_1(\epsilon, \delta)$ samples of $\mathcal{G}$. Each sample is generated by including each edge $e \in E$ with probability p_e. The average pairwise connectivity in the $N_1(\epsilon, \delta)$ sample graphs is computed and returned as an unbiased estimator for $\mathsf{EPC}(\mathcal{G})$.

Algorithm 11. (ϵ, δ) **Monte-Carlo Algorithm to compute** $\mathsf{EPC}(\mathcal{G})$

1. $C_1 \leftarrow 0$.
2. **for** $i = 1$ to $N_1(\epsilon, \delta)$ **do**
   - Draw a sample graph G^i of $\mathcal{G}$.
   - $C_1 = C_1 + \mathcal{P}(G^i)$.
3. Return $\mathcal{E}_1 = \frac{C_1}{N}$ as an unbiased estimator of $\mathsf{EPC}(\mathcal{G})$.

The number of necessary samples to be drawn, denoted by $N_1(\epsilon, \delta)$, is derived based on the following Generalized Zero-One Estimator Theorem introduced by Dagum et al. [28].

Theorem 5.2. *(Generalized Zero-One Estimator [28]) Let* $X_1, X_2, \ldots, X_N$ *be independent identically distributed random variables taking values in* $[0, 1]$*, with mean* $\mu > 0$*. If* $0 < \epsilon < 1$ *and* $N \geq 4(e-2)\ln(2/\sigma)1/(\epsilon^2\mu)$*, where* $e \approx 2.718$ *is Euler's number, then*

$$\Pr\left[(1-\epsilon)\mu \leq \frac{1}{N}\sum_{i=1}^{N} X_i \leq (1+\epsilon)\mu\right] > 1 - \delta.$$

By applying Theorem 5.2 to the i.i.d. random variables $X_i = \mathcal{P}(G^i)/\binom{n}{2}$ with mean $\mu = \mathsf{EPC}(\mathcal{G})/\binom{n}{2}$, we obtain the following lemma.

Lemma 14. *If* $N_1(\epsilon, \delta) \geq 4(e-2)\ln\frac{2}{\sigma}\frac{1}{\epsilon^2\mu}$*, then* $\mathcal{E}_1$ *is an* (ϵ, δ)*-approximation.*

Time complexity. The time to draw a sample and compute the pairwise connectivity is $O(m+n)$. Since we often regard δ as a constant, Algorithm 11 has a time complexity $O\left((m+n)n^2\epsilon^{-2}\mathsf{EPC}(\mathcal{G})^{-1}\right)$.

If $\mathsf{EPC}(\mathcal{G})$ is bounded below by $1/poly(n; m)$, then Algorithm 11 is an FPRAS. The difficult case is the estimation of small values of $\mathsf{EPC}(\mathcal{G})$, where the algorithm is no longer a polynomial-time algorithm. This motivates the construction of better estimation methods to be presented next.

5.2.3 Fully Polynomial Time Approximation Scheme

5.2.3.1 Component Sampling Algorithm

We present an importance sampling method to estimate $\mathsf{EPC}(\mathcal{G})$ in Algorithm 12. In stead of generating the whole sample graph as in Algorithm 11, we select a node $u \in V$ uniformly and perform a Bread-First Search procedure from u, until reaching all nodes in the connected component that contains u. The algorithm then computes the average of the size of the component that contains u less one, and multiply the result by n to obtain an unbiased estimator $\mathcal{E}_2$.

Algorithm 12. (ϵ, δ) **Component Sampling Algorithm to compute** $\mathsf{EPC}(\mathcal{G})$

1. Let $P_E = \sum_{e \in E} p_e$
2. **if** $P_E < 1/m$ **then**
3. return $\mathcal{E}_2 = P_E$.
4. $C_2 \leftarrow 0$.
5. **for** $i = 1$ to $N_2(\epsilon, \delta)$ **do**
   - Select a node $u \in V$ uniformly.
   - Simulate a Breath-First Search from u in $\mathcal{G}$. Let S_i be the number of visited nodes.
   - $C_2 = C_2 + (S_i - 1)$.
6. Return $\mathcal{E}_2 = \frac{nC}{2N}$ as an unbiased estimator of $\mathsf{EPC}(\mathcal{G})$.

Theorem 5.3. *For* $N_2(\epsilon, \delta) = 4(e-2) \ln \frac{2}{\sigma} \frac{n(n-1)}{\epsilon^2 \mathsf{EPC}(\mathcal{G})}$*,* $\mathcal{E}_2$ *is an* (ϵ, δ)*-approximation for* $\mathsf{EPC}(\mathcal{G})$*.*

Proof. In the main loop of Algorithm 12, we can compute S_i with the following equivalent steps: 1) Draw a sample graph G^i; and 2) Select a node u in G^i uniformly and compute S_i as the size of connected component that contains u. Assume that there are t connected components with sizes $s_1, s_2, \ldots, s_k$ in G^i. Then $\mathbb{E}[S_i - 1 | \mathcal{G} = G^i] = \frac{\sum_{i=1}^{k} s_i(s_i-1)}{\sum_{i=1}^{k} s_i} = 2\frac{\mathcal{P}(G^i)}{n}$. Hence $\mathbb{E}[S_i - 1] = 2\mathsf{EPC}(\mathcal{G})/n$.

By applying Theorem 5.2 to i.i.d. $Y_i = (S_i - 1)/(n-1)$ with mean $\mu = \mathsf{EPC}(\mathcal{G})/\binom{n}{2}$, it follows that $\mathcal{E}_2$ is an (ϵ, δ) approximation of $\mathsf{EPC}(\mathcal{G})$. □

Lemma 15. *In any undirected probabilistic graph $\mathcal{G} = (V, E, p)$, we have*

$$\sum_{e \in E} p_e \leq \mathsf{EPC}(\mathcal{G}) \leq \left(1 + \frac{1}{m} \sum_{e \in E} p_e\right)^m.$$

Proof. We prove the lower and upper bounds separately.

Lower bound: By Lemma 12, we have

$$\begin{aligned} \mathsf{EPC}(\mathcal{G}) &= \frac{1}{2} \sum_{u,v \in V; u \neq v} \mathsf{REL}_{uv}(\mathcal{G}) \\ &\geq \sum_{(u,v) \in E} \mathsf{REL}_{uv}(\mathcal{G}) \geq \sum_{(u,v) \in E} p_{uv} \end{aligned}$$

Upper bound: First, we show that $\mathsf{EPC}(\mathcal{G}) \leq \prod_{e \in E}(1 + p_e)$. Then we can apply the inequality of arithmetic and geometric means [20] for positive numbers $(1 + p_e)\ \forall e \in E$ to obtain

$$\mathsf{EPC}(\mathcal{G}) \leq \prod_{e \in E}(1 + p_e) \leq \left(1 + \frac{1}{m} \sum_{e \in E} p_e\right)^m.$$

We prove $\mathsf{EPC}(\mathcal{G}) \leq \prod_{e \in E}(1 + p_e)$ by induction on μ_E the number of *undetermined* edges (those with probabilities strictly less than one).

Basis: If $\mu_E = 0$, we have a deterministic graph with $m = |E|$ edges. Since, the size of the largest component cannot exceed $m + 1$, the pairwise connectivity is at most $1/2n(m+1) < 1/2m(m+1) < 2^m\ \forall m \geq 0$. Thus, the inequality holds for $\mu_E = 0$.

Induction step: Assume that the inequality holds for $\mu_E = t \geq 0$, we show that the inequality also holds when $\mu_E = t + 1$. Assume that $\mu_E = t + 1$, select an arbitrary undetermined edge $(u, v) \in E$ and perform branching on (u, v) as shown in Eq. 5–27. We have

$$\mathsf{EPC}(\mathcal{G}) = p_{uv}\mathsf{EPC}(\mathcal{G}^+) + (1 - p_{uv})\mathsf{EPC}(\mathcal{G}^-),$$

where $\mathcal{G}^+$ is obtained from $\mathcal{G}$ by setting the (u,v)'s probability to one and $\mathcal{G}^-$ is obtained from $\mathcal{G}$ by removing (u,v). Since, both $\mathcal{G}^+$ and $\mathcal{G}^-$ have exactly μ_E undetermined edges, we can apply the induction hypothesis to obtain

$$\begin{aligned}\mathsf{EPC}(\mathcal{G}) &\geq p_{uv}(1+1)\prod_{e\neq(u,v)}(1+p_e)\\ &\quad+(1-p_{uv})\prod_{e\neq(u,v)}(1+p_e)\\ &=(1+p_{uv})\prod_{e\neq(u,v)}(1+p_e)=\prod_{e\in E}(1+p_e).\end{aligned}$$

Thus, the inequality holds for all $\mu_E \geq 0$. □

The bounds in Lemma 15 are asymptotic tight in the sense that there are arbitrary large graphs in which the bounds are only different from the actual values of $\mathsf{EPC}(\mathcal{G})$ by a factor of two. For example, consider $\mathcal{G}$ as a star graph of size n that consists of one center vertex and $n-1$ leaves. All $n-1$ edges are assigned the same probability $1/(n-1)$. One can verify that the lower-bound, $\mathsf{EPC}(\mathcal{G})$, and the upper bound are $1, \frac{3}{2}-\frac{1}{2(n-1)}$, and $\left(1+\frac{1}{n-1}\right)^{n-1}<e$, respectively.

Theorem 5.4. *Algorithm 12 is a fully polynomial randomized approximation scheme (FPRAS) for network connectivity (for any set of edge-dependent failure probabilities* $\{p_e\}_{e\in E}$*.*

Proof. Consider two cases $P_E < n^{-2-\mu}$ and $P_E \geq n^{-2-\mu}$ for some arbitrary small $\mu > 0$.

Case $P_E \leq n^{-2-\mu}$: Let $P_l, l = 0..m$ be the probability that the graph has exactly l edge(s). We have $\sum_{l=0}^{m} P_l = 1$. In addition, let $P_3^+ = \sum_{l=3}^{m} P_l$. We have

$$P_0 = \prod_{e \in E}(1 - p_e) \geq 1 - P_E \tag{5–1}$$

$$P_1 = \sum_{e \in E} p_e \prod_{e\prime \neq e}(1 - p_{e\prime}) = \left(\sum_{e \in E} \frac{p_e}{1 - p_e}\right) P_0 \tag{5–2}$$

$$P_2^+ = 1 - P_0 - P_1 \leq 1 - P_0(1 + \sum_{e \in E} \frac{p_e}{1 - p_e}) \tag{5–3}$$

$$\leq 1 - (1 - P_E)(1 + P_E) = P_E^2 \tag{5–4}$$

We have

$$P_E \leq \mathsf{EPC}(\mathcal{G}) \leq \binom{1}{2} \cdot P_0 + \binom{2}{2} \cdot P_1 + \binom{n}{2} P_2^+ \tag{5–5}$$

$$\leq P_0 \frac{p_E}{1 - p_E} + \binom{n}{2} P_E^2 \tag{5–6}$$

$$\leq \frac{e^{P_E}}{1 - P_E} P_E + o(1) P_E = (1 + o(1)) P_E \tag{5–7}$$

Therefore, P_E is an (ϵ, δ)-approximation for $\mathsf{EPC}(\mathcal{G})$.

Case $P_E \geq n^{-2-\mu}$: From Theorem 3, the component sampling procedure can give an (ϵ, δ) approximation within a polynomial time. □

5.3 Vulnerability Assessment using EPC

In this section, we formulate the vulnerability assessment problems as a mathematical programming problem and devise two approaches to overcome the difficulty of having an exponential number of constraints in the mathematical formulation.

Linear programming for deterministic networks. Given a realization G^l of $\mathcal{G}$, the k-CNP problem in G^l can be formulated as an integer linear programming (ILP),

following [11].

$$\min \sum_{i<j}(1 - x_{ij}) \tag{5–8}$$

$$\text{s. t.} \sum_{i=1}^{n} s_i \leq k$$

$$x_{ij} \leq s_i + s_j + 1 - \xi^{\mathbf{l}}_{\mathbf{ij}}, \quad (i,j) \in E$$

$$x_{ij} + x_{jk} \geq x_{ik}, \quad i,j,k = 1..n$$

$$x_{ij} = x_{ji}, \quad i,j = 1..n$$

$$s \in \{0,1\}^n, x \in \{0,1\}^{n^2} \tag{5–9}$$

where $s_i = 1$ if vertex i is removed and $s_i = 0$, otherwise; x_{ij} represents the "*disconnectivity*" between a pair of nodes i and j after removing $\{i \in V | s_i = 1\}$ i.e. $x_{ij} = 1$ if there is no $i - j$ path and $x_{ij} = 0$, otherwise.

For small networks, the $P(s, x, \xi^l)$ can be solved optimally using branch-and-bound methods. Moreover, further enhancement can be used to enable finding optimal solutions for larger network of several hundreds of nodes. These include the sparse metric methods in [30], that reduces the number of constraints from $O(n^3)$ to $O(mn)$, the removal of the integral conditions on x_{ij} that reduces the number of integral variables from $O(n^2)$ to $O(n)$, and specialized cutting planes [30, 45].

Two-stage stochastic programming. We propose for the k-pCNP problem a two-stage stochastic programming. Stochastic programming has been a common approach for optimization under uncertainty when the probability distribution governs the data is given. A comprehensive introduction to stochastic programming can be found in reference [69]. Our formulation is presented follows with the two highlighted

enhancements from [30].

$$\min_{s\in\{0,1\}^n} \quad \mathbb{E}\left[P(s,x,\xi)\right] \tag{5–10}$$

$$\text{s. t.} \sum_{i=1}^{n} s_i \leq k \tag{5–11}$$

$$\text{where } P(s,x,\xi) = \min \sum_{i<j} (1-x_{ij}) \tag{5–12}$$

$$\text{s. t. } x_{ij} \leq s_i + s_j + 1 - \xi_{\mathbf{ij}},\ (i,j) \in E, \tag{5–13}$$

$$x_{ij} + x_{jk} \geq x_{ik}, \quad (\mathbf{i},\mathbf{j}) \in \mathbf{E}, k = 1..n \tag{5–14}$$

$$x_{ij} = x_{ji}, \quad i,j = 1..n \tag{5–15}$$

$$s \in \{0,1\}^n, x \in [\mathbf{0},\mathbf{1}]^{n^2} \tag{5–16}$$

The optimal decision on which set of vertices to remove only depends on the given topology and the edge probabilities but not the future observations on the edge availabilities. First stage variables s are to be decided before the actual realization of the uncertain parameters in the adjacency matrix ξ. Our objective, the pairwise connectivity in the residual graph, involves only the *expected cost* of x, the variable in the *second stage* (or *recourse*) variables.

Discretization. To solve the stochastic program numerically, one needs to consider all possible realization $G^l \in \mathcal{S}_\mathcal{G}$ and their probability masses $f_\mathcal{G}(G^l)$. Then the two-stage stochastic program can be written as an (one-level) mixed integer programming,

denoted by MIP$_F$:

$$\min \sum_{l=1}^{N} f_{\mathcal{G}}(G^l) \sum_{i<j} (1 - x_{ij}^l) \tag{5–17}$$

$$\text{s. t.} \sum_{i=1}^{n} s_i \leq k \tag{5–18}$$

$$x_{ij}^l \leq s_i + s_j + 1 - \xi_{\mathbf{ij}}^{\mathbf{l}}, (i,j) \in E, l = 1..N \tag{5–19}$$

$$x_{ij}^l + x_{jk}^l \geq x_{ik}^l, \quad (i,j) \in E, k = 1..n, l = 1..N \tag{5–20}$$

$$x_{ij}^l = x_{ji}^l, \quad i,j = 1..n, l = 1..N \tag{5–21}$$

$$s \in \{0,1\}^n, x^l \in [0,1]^{n^2}, \ l = 1..N \tag{5–22}$$

The major challenge in solving this discretized form is that there are exponential number ($N = 2^{|E|}$) of variables and constraints. Thus, solving MIP$_F$ is intractable even for very small instances of $\mathcal{G}$. To overcome this difficulty, we present in next two subsections two approximate mathematical programs of substantially smaller sizes.

5.3.1 Approximating via the Expectation Graph

Many clustering and optimization problem on probabilistic graphs can be reduced into equivalent problems on the (deterministic) *expectation graph*, constructed by casting the edge probabilities into weights. For example, the expectation graph of $\mathcal{G}$ is a (deterministic) graph with the weighted matrix $\overline{\xi}$, where $\overline{\xi}_{ij} = p_{ij}$. The main challenge in this approach is how to interpret the weights in a meaningful way.

Our first approach is to regard the weighted matrix $\overline{\xi}$ as a (binary) adjacency matrix of a deterministic graph. Thus we obtain the following MIP, denoted by MIP$_E$.

$$\min \quad \sum_{i<j}(1-x_{ij}) \tag{5–23}$$

$$\text{s. t.} \quad \sum_{i=1}^{n} s_i \leq k \tag{5–24}$$

$$x_{ij} \leq s_i + s_j + 1 - \overline{\xi}_{\mathbf{ij}}, \quad (i,j) \in E \tag{5–25}$$

$$Constraints\ (5--14), (5--15), \& \ (5--16)$$

Not only this relaxation has polynomial time numbers of constraints and variables, but also its optimal solution provides a lower-bound on the optimal solution of MIP$_F$, as stated in the following lemma.

Lemma 16. *MIP$_E$ is a mixed integer programming with at most n integral variables and $O(mn)$ constraints. Moreover, the objective of the MIP$_E$ is at most that of the MIP$_F$.*

Proof. The number of integral variables and constraints can be proven similar to [30].

To show that the the objective of the MIP$_E$ is a lower-bound on that of the MIP$_F$, we construct a feasible solution $(\tilde{s}, \tilde{x})$ of MIP$_E$ that gives an objective equal to the optimal objective of MIP$_F$.

Let $\left(\hat{s}, \hat{x}^1, \ldots, \hat{x}^N\right)$ be an optimal solution of the MIP$_F$. Construct a solution $\left(\tilde{s} = \hat{s}, \tilde{x} = \sum_{l=1}^{N} f_{\mathcal{G}}(G^l)\hat{x}^l\right)$. The objective value of MIP$_E$ given by that solution is

$$\begin{aligned}\sum_{i<j}(1-\tilde{x}_{ij}) &= \sum_{i<j}(1-\sum_{l=1}^{N} f_{\mathcal{G}}(G^l)\hat{x}^l_{ij}) \\ &= \sum_{i<j}\sum_{l=1}^{N} f_{\mathcal{G}}(G^l)(1-\hat{x}^l_{ij})\end{aligned}$$

which is exactly the optimal objective of MIP$_F$. The last equality holds because the probabilities $f_{\mathcal{G}}(G^l)$ add up to one.

The rest is to show that $(\tilde{s}, \tilde{x})$ is a feasible solution of MIP$_E$. Clearly, $\tilde{s}$ satisfy (5–24) and the integral constraints. Also since $\tilde{x}$ is a convex combination of $\hat{x}^l, l = 1..N$ with the masses $f_{\mathcal{G}}(G^l)$, $\tilde{x}$ satisfy the constraints (5–25), (5–14), (5–15), & (5–16) as they can be inferred from the same convex combination of the constraints (5–19) to (5–22). □

One can solve MIP$_E$ optimally using the branch-and-cut method in [30] to obtain 1) a set of k critical nodes and 2) a lower-bound on the minimum expected pairwise connectivity after removing k nodes. We note that the non-integrality of x_{ij} is essential for MIP$_E$. When x_{ij} is restricted to $\{0, 1\}$, e.g. in (5–9), the constraints (5–25) is essentially $x_{ij} \leq s_i + s_j$ and MIP$_E$ become equivalent to IP (5–8)-(5–9). That is the information encoded in the edge probabilities is disregarded and only the network topology is used in the formulation. In addition, since the convex combination of $\hat{x}^l$ is a fractional vector, we will not be able to derive the lower-bound given in Lemma 16.

In large networks, branch-and-cut algorithm starts to show its exponentially running time, the following randomized rounding algorithm can be used to obtain a set of k critical nodes. The rounding procedure is described in Algorithm 11. The algorithm repeatedly solves an LP relaxation of MIP$_E$ and round up the maximum s_i to one, provided that s_i is not rounded before. After k steps, k nodes that have $s_i = 1$ are retuned as the set of the critical nodes. Since the LP relaxation has at most $O(mn)$ constraints solving the LP relaxation takes an $O(m^3n^3)$ time [30] in the worst case. Thus the total time complexity is at most $O(km^3n^3)$.

Algorithm 13. Rounding on the Expectation Graph Algorithm (REGA)

1. Obtain an LP relaxation of MIP$_E$ with the relaxed constraints $s \in [0,1]^n$.
2. Initialize the set of selected nodes $D = \emptyset$.
3. Repeat k times the following steps
   - Solve the LP relaxation
   - Select $u = \arg\max \in V \setminus Ds_i$.
   - Add u to D and fix $s_u = 1$
4. Return k critical nodes in D.

5.3.2 Sample Average Approximation (SAA) Method

Our second approach to reduce the number of realizations is to apply the Sample Average Approximation (SAA) method. We generate independently T samples $\xi^1, \xi^2, \cdots, \xi^T$ using Monte Carlo simulation (i.e. to generate each edge $(u,v) \in e$ with probability p_{uv}). The expectation objective $q(s) = \mathbb{E}[P(s,x,\xi)]$ is then approximated by the sample average $\hat{q}_T(x) = \frac{1}{T}\sum_{l=1}^{T}\sum_{i<j}(1 - x_{ij}^l)$, and the new formulation is then

$$\begin{aligned} \min \quad & \tfrac{1}{T}\textstyle\sum_{l=1}^{T}\sum_{i<j}(1 - x_{ij}^l) \\ \text{s. t.} \quad & \text{Constraints}(5 - -18) - (5 - -22), \text{replacing } N \text{ with } T \end{aligned}$$

Under some regularity conditions $\frac{1}{T}\sum_{j=1}^{T}\sum_{i<j}(1 - x_{ij}^l)$ converges pointwise with probability 1 to $\mathbb{E}[P(s,x,\xi)]$ as $T \to \infty$. Moreover, an optimal solution of the sample average approximation provides an optimal solution of the stochastic programming with probability approaching one exponentially fast w.r.t. T. Formally, denote by s^* and $\hat{s}$ the optimal solution of the stochastic programming and the sample average approximation,

respectively. For any $\epsilon > 0$, it can be derived from Propostition 2.2 in [52] that

$$\begin{aligned} &\Pr\left[\mathbb{E}\left[P(s,\hat{x},\xi)\right] - \mathbb{E}\left[P(s,x^*,\xi)\right] > \epsilon\right] \\ &\qquad \leq \exp\left(-T\frac{\epsilon^2}{n^4} + n\log k\right) \end{aligned} \tag{5–26}$$

Equivalently, if $T \geq \frac{n^4}{\epsilon^2}(nk - \log\alpha)$, then $\Pr\left[\mathbb{E}\left[P(s,\hat{x},\xi)\right] - \mathbb{E}\left[P(s,x^*,\xi)\right] < \epsilon\right] > 1-\alpha$ for any $\alpha \in (0,1)$. Although the estimation on T maybe too conservative for practical estimates, it is expected that the optimal value and optimal solutions of the SAA problem converge to their counterparts even with a reasonable small value of T. The description for SAA method is summarized in Algorithm 14. The algorithm consists of two phases. In the first phase, the delayed constraints technique is used to incrementally construct and solve an LP relaxation of the SAA. In the second phase, the same iterative rounding procedure in Algorithm 1 is applied to find k critical nodes by rounding up the fractional solution.

Algorithm 14. Sample Ave. Approx. Algorithm (SA3)

Parameter T: the number of sampling

Phase 1: Delayed Constraints

1. Initialize an LP with the objective $\frac{1}{T}\sum_{l=1}^{T}\sum_{i<j}(1-x_{ij}^{l})$ and only the constraints $s \in [0,1]^n, x_{ij}^{l} \in [0,1]$.
2. **for** $l = 1..T$ **do**
   - Generate the l^{th} sample of $\mathcal{G}$ (adjacency matrix ξ^{l}).
   - Add the constraints involved x_{ij}^{l} to the LP.
   - Solve the updated LP.

Phase 2: Iterative rounding

3. Initialize the set of selected nodes $D = \emptyset$.
4. Repeat k times the following steps
   - Select $u = \arg\max_{i \in V \setminus D} s_i$.
   - Add u to D and fix $s_u = 1$
   - Re-solve the LP
5. Return k critical nodes in D.

5.3.3 Local Search Heuristic

A local search method described in Algorithm 15 to find the k critical nodes. The algorithm selects k nodes in a greedy manner: each time the algorithm selects the node that removal results in the largest degradation in terms of EPC. Moreover, the algorithm attempts to swap a node w outside the disruptor with a node u in the disruptor that gives the sharpest decrease in EPC. The local search terminates when no improvement exists.

Proof of Lemma 12

Algorithm 15. Iterative Greedy Algorithm (IGA)

1: $D \leftarrow \emptyset$
2: **for** $i = 1..k$ **do**
3: $\quad u = \arg\min_{v \in V \setminus S} \mathsf{EPC}(G_{[V \setminus (S \cup \{v\})]})$
4: $\quad D \leftarrow D + \{u\}$
5: **while** $\exists (u, v) \in D \times (V \setminus D)$
6: $\quad$ & (swapping u, v decreases the objective) **do**
7: $\quad D \leftarrow D + \{v\} - \{u\}$
8: Output D.

Proof. This is derived directly from the definition of $\mathsf{EPC}(\mathcal{G})$. Define $\mathsf{conn}_{uv}(G^l) = 1$, if there is a path between u and v in a sample graph G^l and $\mathsf{conn}_{uv}(G^l) = 0$, otherwise. We have

$$\begin{aligned}\mathsf{EPC}(\mathcal{G}) &= \sum_{l=1}^{N} f_{\mathcal{G}}(G^l)\mathcal{P}(G^l) = \sum_{i=1}^{N} f_{\mathcal{G}}(G^l)\frac{1}{2}\sum_{u \neq v}\mathsf{conn}_{uv}(G^l) \\ &= \frac{1}{2}\sum_{u \neq v}\sum_{l=1}^{N} f_{\mathcal{G}}(G^l)\mathsf{conn}_{uv}(G^l) = \frac{1}{2}\sum_{u \neq v}\mathsf{REL}_{u,v}(\mathcal{G})\end{aligned}$$

□

Proof of Lemma 13

Proof. An edge $(u, v) \in E_B$ is said to be *undetermined*, if $0 < p^A_{uv} p^B_{uv} < 1$. We prove the lemma by induction on the number of *undetermined* edges within E_B, denoted by μ_B.

Basis: If $\mu_B = 0$, both $\mathcal{G}_A$ and $\mathcal{G}_B$ are deterministic graph, and the statement holds trivially. Assume that the lemma is true when $\mu_B = t \geq 0$. We show that the lemma is also true when $\mu_B = t + 1$.

Induction step: Assume that $\mu_B = t + 1$, pick an arbitrary edge $(u, v) \in E_B$ and consider the branching on (u, v):

$$\mathsf{EPC}(\mathcal{G}_A) = p^A_{uv}\mathsf{EPC}(\mathcal{G}^+_A) + (1 - p^A_{uv})\mathsf{EPC}(\mathcal{G}^-_A). \tag{5–27}$$

where $\mathcal{G}_A^+$ is obtained from $\mathcal{G}_A$ by assigning $p_{uv} = 1$; and $\mathcal{G}_A^-$ is obtained from $\mathcal{G}_A$ by removing the edge (u, v).

Similarly, we have

$$\mathsf{EPC}(\mathcal{G}_B) = p_{uv}^B \mathsf{EPC}(\mathcal{G}_B^+) + (1 - p_{uv}^B)\mathsf{EPC}(\mathcal{G}_B^-).$$

Since $\mathcal{G}_A \preceq \mathcal{G}_B$, it can be verified that $\mathcal{G}_A^+ \preceq \mathcal{G}_B^+$ and $\mathcal{G}_A^- \preceq \mathcal{G}_B^-$. Note that the pairs $(\mathcal{G}_A^+, \mathcal{G}_B^+)$ and $(\mathcal{G}_A^-, \mathcal{G}_B^-)$ have at most t undetermined edges. By the induction hypothesis, we have

$$\begin{aligned}\mathsf{EPC}(\mathcal{G}_B) &\geq p_{uv}^B \mathsf{EPC}(\mathcal{G}_A^+) + (1 - p_{uv}^B)\mathsf{EPC}(\mathcal{G}_A^-) \\ &\geq p_{uv}^A \mathsf{EPC}(\mathcal{G}_A^+) + (1 - p_{uv}^A)\mathsf{EPC}(\mathcal{G}_A^-) = \mathsf{EPC}(\mathcal{G}_A).\end{aligned}$$

The last inequality holds because $p_{uv}^A \geq p_{uv}^N$ and $\mathsf{EPC}(\mathcal{G}_A^+) \geq \mathsf{EPC}(\mathcal{G}_A^-)$ which can be shown based on the fact that each sample of graph $\mathcal{G}_A^+$ can be generated by first generating a sample of $\mathcal{G}_A^-$ and then add (u, v) the sample. Obviously, adding an edge to a (deterministic) graph will not decrease the pairwise connectivity. Thus, the lemma holds for all $\mu_B \geq 0$. □

CHAPTER 6
CASCADING-FAILURES IN NETWOKRS

In this chapter, we formulate the measuring vulnerabiltiy in the presence of cascading-failure as an optimization problem: the Cost-effective, massive and outbreak problem (CFM). In Section 6.1, we analyze the propagation process on power-law networks to give an lower-bound on the seeding size. We present VirAds, a scalable algorithm to find a minimal seeding for the CFM problem in Section 6.2. The hardness of finding a cost-effective seeding is addressed in Section 6.3. Finally, we perform extensive experiments on large social networks such as Facebook and Orkut to confirm the efficiency of our proposed algorithm and analyze the results to give new observations to information diffusion process in networks.

6.1 Seeding Cost of Massive Outbreak

In this section, we exploit the power-law topology found in most complex networks [12, 13, 25] to demonstrate that when the propagation hop is limited, a large number of seeding nodes is needed to spread the influence throughout the network. The size of seeding is proved to be a constant fraction of the number of vertices n, which is prohibitive for large social networks of millions of nodes. We first summarize the well-known power-law model in [3]; then we use the model to prove the prohibitive seeding cost for the CFM problem.

6.1.1 Power-law Network Model.

Many complex systems of interest including OSNs are found to have the degree distributions approximately follows the power laws [12, 13, 25]. That is the fraction of nodes in the network having k connections to other nodes is proportional to $k^{-\gamma}$, where γ is a parameter whose value is typically in the range $2 < \gamma < 3$. Those networks have been used in studying different aspects of the scale-free networks [3, 5, 39, 41]. We follow the $P(\alpha, \gamma)$ power-law model in [3] in which the number of vertices of degree k

is $\lfloor \frac{e^\alpha}{k^\gamma} \rfloor$ where e^α is the normalization factor. For convenience, we shall refer to such a network as a $P(\alpha, \gamma)$ network.

We can deduce that the maximum degree in a $P(\alpha, \gamma)$ network is $e^{\frac{\alpha}{\gamma}}$ (since for $k > e^{\frac{\alpha}{\gamma}}$, the number of edges will be less than 1). The number of vertices and edges are

$$n = \sum_{k=1}^{e^{\frac{\alpha}{\gamma}}} \frac{e^\alpha}{k^\gamma} \approx \begin{cases} \zeta(\gamma)e^\alpha & \text{if } \gamma > 1 \\ \alpha e^\alpha & \text{if } \gamma = 1 \\ \frac{e^{\frac{\alpha}{\gamma}}}{1-\gamma} & \text{if } \gamma < 1 \end{cases},$$

$$m = \frac{1}{2}\sum_{k=1}^{e^{\frac{\alpha}{\gamma}}} k\frac{e^\alpha}{k^\gamma} \approx \begin{cases} \frac{1}{2}\zeta(\gamma-1)e^\alpha & \text{if } \gamma > 2 \\ \frac{1}{4}\alpha e^\alpha & \text{if } \gamma = 2 \\ \frac{1}{2}\frac{e^{\frac{2\alpha}{\gamma}}}{2-\gamma} & \text{if } \gamma < 2 \end{cases} \tag{6–1}$$

where $\zeta(\gamma) = \sum_{i=1}^{\infty} \frac{1}{i^\gamma}$ is the Riemann Zeta function [3] which converges for $\gamma > 1$ and diverges for all $\gamma \leq 1$. Without affecting the conclusion, we will simply use real numbers instead of rounding down to integers. The error terms are sufficiently small and can be bounded in our proofs.

While the scale of the network depends on α, the parameter γ decides the connection pattern and many other important characterizations of the network. For instance, the larger γ, the sparser and the more "power-law" the network is. Hence, the parameter γ is often regarded as the characteristic constant for scale-free networks.

6.1.2 Prohibitive Seeding Costs

We prove that the seeding must contain at least $\Omega(n)$ vertices if the propagation is locally bounded. The result is stated in the following theorem.

Theorem 6.1. *Given a power-law network $G \in P(\alpha, \gamma)$, with $\gamma > 2$ and constant $0 < \rho < 1$, any d-seeding is of size at least $\Omega(n)$.*

Proof. The proof consists of two parts. In the first part, we show that the volume i.e. the total degree of vertices, of any d-seeding must be $\Omega(m)$. In the second part, we prove

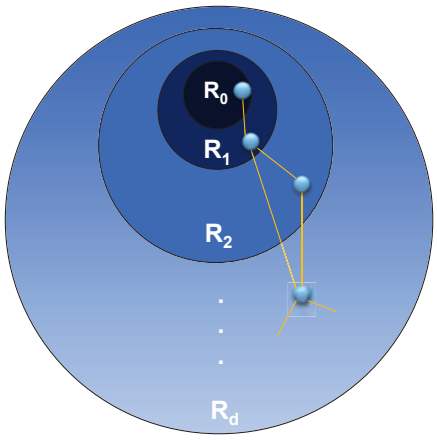


Figure 6-1. The influence propagation in the network.

that any subset of vertices $S \subset V$ with volume $\mathrm{vol}(S) = \Omega(m)$ in a power-law network with power-law exponent $\gamma > 2$, will imply that $|S| = \Omega(n)$. Thus, the theorem follows.

In the first part, we consider two separate cases

Case $\rho > \frac{1}{2}$: Let $S = R_0$ be the optimal solution for the CFM problem on $G = (V, E)$, and $S = R_0, R_1, R_2, \ldots, R_d$ are vertices that become active at round $0, 1, 2, \ldots, d$, respectively (see Fig. 6-4). Notice that $\{R_i\}_{i=0}^{d}$ form a partition of V. Moreover, for each $1 \leq t \leq d$ the following inequality holds.

$$|\phi(R_t, \bigcup_{i=0}^{t-1} R_i)| \geq \frac{\rho}{1-\rho} \left(|\phi(R_t, \bigcup_{j=t+1}^{d} R_j)| + 2|\phi(R_t, R_t)| \right) \tag{6–2}$$

where $\phi(A, B)$ denotes the set of edges connecting one vertex in A to one vertex in B. The inequality means that at least a fraction $\frac{\rho}{1-\rho}$ among edges incident with the vertices activated in round t must be incident with active vertices in the previous rounds.

Sum up all inequalities in (6–2) for $t = 1..d$, we have

$$\sum_{t=1}^{d} |\phi(R_t, \bigcup_{i=0}^{t-1} R_i)| \geq \frac{\rho}{1-\rho} \sum_{t=1}^{d} \left(|\phi(R_t, \bigcup_{j=t+1}^{d} R_j)| + 2|\phi(R_t, R_t)| \right)$$

Eliminate the common factors in both sides, we have

$$\sum_{i=0}^{d-1} |\phi(R_i, \bigcup_{t=i+1}^{d} R_t)| \geq \frac{\rho}{1-\rho} \sum_{j=1}^{d-1} |\phi(R_j, \bigcup_{t=j+1}^{d} R_t)| + 2\sum_{t=1}^{d-1} |\phi(R_t, R_t)|$$

After some algebra, we obtain

$$\begin{aligned} \mathrm{vol}(R_0) &\geq |\phi(R_0, \bigcup_{t=1}^{d} R_t)| \\ &\geq \frac{2\rho-1}{1-\rho} \sum_{j=1}^{d-1} |\phi(R_i, \bigcup_{t=j+1}^{d} R_t)| + 2\sum_{t=1}^{d} |\phi(R_t, R_t)| \\ &\Leftrightarrow \frac{\rho}{1-\rho} |\phi(R_0, V)| - |\phi(R_0, R_0)| \\ &\geq \frac{2\rho-1}{1-\rho} |E| + \frac{3-4\rho}{1-\rho} \sum_{t=1}^{d} |\phi(R_t, R_t)| \end{aligned} \tag{6–3}$$

Hence, when $\rho > 1/2$, $\mathrm{vol}(R_0) \geq \frac{2\rho-1}{1-\rho}|E| = \Omega(m)$ for any d-seeding R_0.

Case $\rho \leq \frac{1}{2}$: We say that an edge is active if it is incident to at least one active vertex. At round $t = 0$, there are at most $\mathrm{vol}(R_0)$ active edges, those who are incident to R_0. Eq. 6–2 implies that the number of active edges in each round increases at most ρ^{-1} times. After d rounds, the number of active edges will be bounded by $\mathrm{vol}(R_0) \times \rho^{-d}$. Since, all edges are active at the end we have the inequality:

$$\mathrm{vol}(R_0) \geq \rho^{-d}|E|.$$

In the second part of the proof, we show that if a subset $S \subset V$ has $\mathrm{vol}(S) = \Omega(m)$, then $|S| = \Omega(n)$ whenever the power-law exponent $\gamma > 2$. Assume that $\mathrm{vol}(S) \geq cm$, for some positive constant c. The size of S is minimum when S contains only the highest degree vertices of V. Let k_0 be the minimum degree of vertices in S in that extreme case, by Eq.

6–1 we have

$$cm = \frac{c}{2}\sum_{k=1}^{e^{\frac{\alpha}{\gamma}}} k\frac{e^{\alpha}}{k^{\gamma}} \leq \mathrm{vol}(S) \leq \frac{1}{2}\sum_{k=k_0}^{e^{\frac{\alpha}{\gamma}}} k\frac{e^{\alpha}}{k^{\gamma}}$$

Simplify two sides, we have

$$\sum_{k=1}^{k_0-1}\frac{1}{k^{\gamma-1}} \leq (1-c)\sum_{k=1}^{e^{\frac{\alpha}{\gamma}}}\frac{1}{k^{\gamma-1}} = (1-c)\zeta(\gamma-1)$$

Since, the zeta function $\zeta(\gamma-1)$ converges for $\gamma > 2$, there exists a constant $k_{\rho,\gamma}$ that depends only on ρ and γ that satisfies

$$\sum_{k=1}^{k_{\rho,\gamma}}\frac{1}{k^{\gamma-1}} > (1-c)\zeta(\gamma-1)$$

Obviously, we have $k_0 \leq k_{\rho,\gamma}$. Thus, the number of vertices that are in S is at least

$$\sum_{k=k_{\rho,\gamma}}^{e^{\frac{\alpha}{\gamma}}}\frac{e^{\alpha}}{k^{\gamma}} = (1-\sum_{k=1}^{k_{\rho,\gamma}}\frac{1}{k^{\gamma}})n = \Omega(n)$$

We have the last step because the sum $\sum_{k=1}^{k_{\rho,\gamma}}\frac{1}{k^{\gamma}}$ is bounded by a constant since $k_{\rho,\gamma}$ is a constant. □

In both cases $\rho > 1/2$ and $\rho \leq 1/2$, the size of a d-seeding set is at least $\Omega(n)$. However, we can see a clear difference in the propagation speed with respect to d between two cases. When $\rho < 1/2$, the number of active edges can increase exponentially (but is still bounded if d is a constant) and, it is likely that the number of active vertices also exponentially increases. In contrast, when $\rho > 1/2$, exploding in the number of active edges (and hence active vertices) is impossible as the volume of the d-seeding is tied to the number of edges m by a fixed constant $\frac{2\rho-1}{1-\rho}$, regardless of the value of d.

6.2 Algorithm to Identify the Minimum Outbreak Seeding

In order to understand the influence propagation when the number of propagation hops is bounded, we propose VirAds, an efficient algorithm for the CFM problem. With

the huge magnitude of OSN users and data available on OSNs, scalability becomes the major problem in designing algorithm for CFM. VirAds is scalable to network of hundred of millions links and provides high quality solutions in our experiments.

Before presenting VirAds, we consider a natural greedy for the CFM problem in which the vertex that can activate the most number of inactive vertices within d hops is selected in each step. This greedy is unlikely to perform well on practice for following two reasons. First, at early steps, when not many vertices are selected, every vertex is likely to activate only itself after being chosen as a seed. Thus, the algorithm cannot distinguish between good and bad seeds. Second, the algorithm suffers serious scalability problems. To select a vertex, the algorithm has to evaluate for each vertex v how many vertices will be activated after adding v to the seeding, e.g. by invoking an $O(m+n)$ Breadth-First Search procedure rooted at v. In the worst-case when $O(n)$ vertices are needed to evaluate, this alone can take $O(n(m+n))$. Moreover, as shown in the previous section, the seeding size can be easily $\Omega(n)$; thus, the worst-case running time of the naive greedy algorithm is $O(n^2(m+n))$, which is prohibitive for large-scale networks.

As shown in Algorithm 16, our VirAds algorithm overcomes the mentioned problems in the naive greedy by favoring the vertex which can activate the most number of *edges* (indeed, it also considers the number of active neighbor around each vertex). This avoids the first problem of the naive greedy algorithm. At early steps, the algorithm behaves similar to the degree-based heuristics that favors vertices with high degree. However, when a certain number of vertices are selected, VirAds will make the selection based on the information within d-hop neighbor around the considered vertices rather than only one-hop neighbor as in the degree-based heuristic.

The scalability problem is tackled in VirAds by efficiently keeping track of the following measures for each vertex v.

- $\mathbf{r_v}$: the round in which v is activated

Algorithm 16: VirAds: Finding Influence Nodes in Networks

Input: Graph $G = (V, E)$, $0 < \rho < 1$, $d \in \mathbb{N}^+$
Output: A small d-seeding
$n_v^{(e)} \leftarrow d(v), n_v^{(a)} \leftarrow \rho \cdot d(v), r_v \leftarrow d + 1, v \in V$;
$r_v^{(i)} = 0, i = 0..d, P \leftarrow \emptyset$;
while *there exist inactive vertices* **do**
  repeat
    $u \leftarrow \text{argmax}_{v \notin P} \{n_v^{(e)} + n_v^{(a)}\}$;
    Recompute $n_v^{(e)}$ as the number of new active edges after adding u.
  until $u = \textit{argmax}_{v \notin P} \{n_v^{(e)} + n_v^{(a)}\}$;
  $P \leftarrow P \cup \{u\}$;
  Initialize a queue: $Q \leftarrow \{(u, r_v)\}$;
  $r_u \leftarrow 0$;
  foreach $x \in N(u)$ **do**
    $n_x^{(a)} \leftarrow \max\{n_x^{(a)} - 1, 0\}$;
  while $Q \neq \emptyset$ **do**
    $(t, \tilde{r}_t) \leftarrow Q.pop()$;
    foreach $w \in N(t)$ **do**
      foreach $i = r_t$ *to* $\min\{\tilde{r}_t - 1, r_w - 2\}$ **do**
        $r_w^{(i)} = r_w^{(i)} + 1$;
        if $(r_w^{(i)} \geq \rho \cdot d_w) \wedge (r_w \geq d) \wedge (i + 1 < d)$ **then**
          foreach $x \in N(w)$ **do**
            $n_x^{(a)} \leftarrow \max\{n_x^{(a)} - 1, 0\}$;
          $r_w = i + 1$;
          if $w \notin Q$ **then**
            $Q.push((w, r_w))$;
  Output P;

- $\mathbf{n_v^{(e)}}$: The number of new active edges after adding v into the seeding
- $\mathbf{n_v^{(a)}}$: The number of extra active neighbors v needs in order to activate v
- $\mathbf{r_v^{(i)}}$: The number of activated neighbors of v up to round i where $i = 1..d$.

Given those measures, VirAds selects in each step the vertex u with the highest *effectiveness* which is defined as $n_u^{(e)} + n_u^{(a)}$. After that, the algorithm needs to update the measures for all the remaining vertices.

Except for $n_v^{(e)}$, we show that all other measures can be effectively kept track of in only $O((m+n)d)$ during the whole algorithm. When a vertex u is selected, it causes a chain-reaction and activate a sequence of vertices or lower the rounds in which vertices are activated. New activated vertices together with their active rounds are successively pushed into the queue Q for further updating much like what happens in the Bellman-Ford shortest-paths algorithm. Everytime we pop a vertex v from Q, if r_v, the current active round of v, is different from $\tilde{r}_v$, the active round of v when v is pushed into Q, we update for each neighbor w of v the values of r_w and $r_w^{(i)}$. If any neighbor w of v changes its active round and w is not in Q, we push w into Q for further update. The update process stops when Q is empty. Note that for each node $u \in V$, changing of r_u can cause at most d update for $r_w^{(.)}$ where w is a neighbor of u. For all neighbors of u, the total number of update is, hence, $O(d \cdot d(u))$. Thus, the total time for updating $r_w^{(.)}\ \forall w \in V$ in VirAds will be at most $O((m+n) \cdot d)$.

To maintain $n_v^{(e))}$, the easiest approach is to recompute all $n_v^{(e)}$. This approach, called *Exhaustive Update*, is extremely time-consuming as discussed in the naive greedy. Instead, we only update $n_v^{(e)}$ when "necessary". In details, vertices are stored in a max priority queue in which the priority is their *effectiveness*. In each step, the vertex u with the highest effectiveness is extracted and $n_u^{(e)}$ is recomputed. If after updating, u still has the highest effectiveness, u is then selected. Otherwise, u is pushed back to the priority queue, and the new vertex with the highest effectiveness is considered, and so on.

Approximation Ratio for Power-law Networks.

The CFM problem can be easily shown to be NP-hard by a reduction from the set cover problem. Thus, we are left with two choices: designing heuristics which have no worst-case performance guarantees or designing approximation algorithms which can guarantee the produced solutions are within a certain factor from the optimal. Formally,

a β- approximation algorithm for a minimization (maximization) problem always returns solutions that are at most β times larger (smaller) than an optimal solution.

Unfortunately, there is unlikely an approximation algorithm with factor less than $O(\log n)$ as shown in next section. However, if we assume the network is power-law, our VirAds is an approximation algorithm for CFM with a constant factor.

Theorem 6.2. *In power-law networks, VirAds is an $O(1)$ approximation algorithm for the CFM problem for bounded value of d.*

The theorem follows directly from the result in previous section that the optimal solution has size at least $\Omega(n)$ in power-law networks. Thus, the ratio between the VirAds's solution and the optimal solution is bounded by a constant.

6.3 Hardness of the CFM Problem

This section provides the hardness of approximating the optimal solutions of the CFM problem, the impossibility of finding near-optimal solutions in polynomial time. In previous Section, we can obtain $O(1)$ approximation algorithms for CFM when the network is power-law. However, without the power-law assumption, there is no algorithm that can approximate the problem within a factor less than $O(\log n)$. We first prove the hardness for the case when $d = 1$, which is an essential step in proving the hardness for the general case $d \geq 1$. We begin with the Feige's reduction for proving $\ln n$ threshold for the set cover problem. Our proof for the hardness of approximation for the CFM problem requires understanding the Feige's construction together with its parameter settings.

6.3.1 Feige's Reduction for Set Cover

Feige presented a reduction from a k-prover proof system for a MAX 3SAT-5 instance ϕ that is a *conjunctive normal form* formula consists of n variables and $\frac{5n}{3}$ clauses of exactly 3 literals. The verifier interacts with k provers, and ask provers different questions based on a random string r; each question involves $l/2$ clauses and $l/2$ variables. If the formula ϕ is satisfiable, then the provers have a strategy that cause the verifier accepts for all random strings. If only a $(1 - \epsilon)$ fraction of the clauses in ϕ

are simultaneously satisfiable, then for all strategies of the provers, the verifier weakly accept with a probability at most $k^2 \cdot 2^{-cl}$, where c is a constant that depends only on ϵ.

The core of the Set cover gadget is a partition system $B(m, L, k, d)$, where B is a ground set of m points. The partition system is a collection of $L = 2^l$ partitions $P_1, \ldots, P_L$ of B, each partition P_i has exactly k disjoint subsets $p_{i,1}, \ldots, p_{i,k}$. Any cover of m points in B requires at least $d = (1 - \frac{2}{2})k \ln m$ subsets. The condition to make constructing such a system possible is that $k < \frac{\ln m}{3 \ln \ln m}$.

Let $R = (5n)^l$ denote the number of possible random strings for the verifier. We make R copies of partition system B. Let B_r denote the copy of the partition associated with the random string r and $p^r_{i,j}$ the copy of set $p_{i,j}$ in B_r.

We now ready to describe the instance of Set Cover in the Feige's reduction. The universal set $\mathcal{U} = \bigcup_{r \in R} B_r$ contains $N = |\mathcal{U}| = mR$ points; and the set system is $\mathcal{S} = \{S_{q,a,i}\}_{q,a}$, where i can be deduced from syntax of (q, a). Each set $S_{q,a,i}$ corresponds to a question-answer pair (q, a) of the ith prover and $S_{q,a,i} = \bigcup_{(q,i) \in r} p^r_{a_r,i}$ where $(q, i) \in r$ means on random string r, the ith prover receives question q, and a_r is the assignment of variables extracted from a.

As long as $k^2 2^{-cl} < \frac{8}{k^3 \ln^2 m}$, we obtain the hardness result $(1 - \frac{4}{k}) \ln m$ i.e. if formula ϕ is satisfiable, then mR points in $\mathcal{U}$ can be covered by kQ subsets, and if only $(1 - \epsilon)$ fraction of the clauses are simultaneously satisfiable, the minimum set cover has size at least $(1 - \frac{4}{k}) \ln m \; kQ$. Here, Q is the set of all $n^l \, (5/3)^{l/2}$ possible questions. The condition can be satisfied with $l > \frac{1}{c} (5 \log k + 2 \log \ln m)$.

The hardness ratio $(1 - f(k)) \ln m$ of the set cover is obtained from the following key lemma.

Lemma 17. *(Lemma 4.1 [38]) If ϕ is satisfiable, then the above set of $N = mR$ points can be covered by kQ subsets. If only a $(1 - \epsilon)$ fraction of the clauses in ϕ are simultaneously satisfiable, the above set requires $(1 - 2f(k))kQ \ln m$ subsets in order to be covered, where $f(k) \to 0$ as $k \to \infty$.*

Note that $\ln m = (1-\epsilon)\ln N$ by the setting of n, l, and m in the proof. Thus, the final hardness ratio is $(1-\epsilon)\ln N$, where $N = |\mathcal{U}|$. However, we can choose different settings of n, l, and m and obtain different hardness ratios.

We finish the present of Feige's reduction by giving upper bounds for quantities that appear later in our proofs.

- The *number of subsets* $|\mathcal{S}| \leq |Q|2^{2l}$. Since, for each question $q \in Q$, there are at most 2^{2l} answers of $2l$ bit length.
- The *maximum size* of a subset $\Delta_{\mathcal{S}} = \max_{S \in \mathcal{S}} |S| \leq m3^{l/2}$. Since each i and $q \in Q$ there are at most $3^{l/2}$ random strings r such that the verifier makes query q to the ith prover and $|p^r_{a_r,i}| \leq m$.
- The *maximum frequency* of a point (element) in $\mathcal{U}$: $f \leq k2^l$. Because, for a pair (q,i), each partition $p^r_{a_r,i}$ is included at most 2^l times, plus each point in B_r appears in exactly k partitions.

6.3.2 One-hop CFM

We prove that the CFM problem cannot be approximated within a factor $\ln \Delta - O(\ln \ln \Delta)$ in graphs of maximum degree Δ, unless P=NP. The proof uses a gap-reduction from an instance of the *Bounded Set Cover* problem (SC_B) to an instance of CFM problem whose degrees are bounded by $B' = B$ poly log B. For background on hardness of approximation and gap-reduction we refer to reference [8].

Definition 8 (Bounded Set Cover)**.** *Given a set system $(\mathcal{U}, \mathcal{S})$, where $\mathcal{U} = \{e_1, e_2, \ldots, e_{n_s}\}$ is a universe and $\mathcal{S}$ is a collection of subsets of $\mathcal{U}$. Each subset in $\mathcal{S}$ has at most B elements and each element belongs to at most B subsets, for a predefined constant $B > 0$. A cover is a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of sets whose union is $\mathcal{U}$. Find a cover which uses the minimum number of subsets.*

We state the tight inapproximability result for the bounded set cover by Trevisan [74] in the following lemma.

Lemma 18. *There exist constants $B_0, c_0 > 0$ such that for every $B \geq B_0$ it is NP-hard to approximate the SC_B problem within a factor of $\ln B - c_0 \ln \ln B$.*

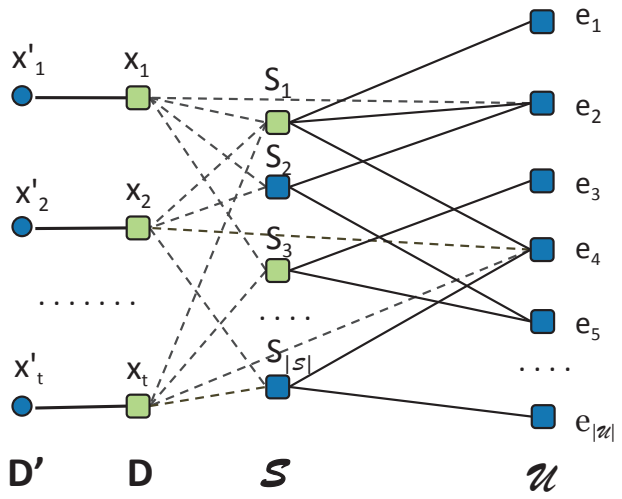


Figure 6-2. Reduction from SC_B to CFM when $d = 1$

The proof in [74] reduces an instance of $GAP - SAT_{1,\gamma}$ of size n_S to an instance $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ of SC_B by settings parameters l, m in Feige's construction [38] to be $\theta(\ln \ln B)$ and $\frac{B}{\text{poly log}(B)}$, respectively. Denote by Δ_S the maximum cardinality of sets, and by f the maximum frequency of elements in $\mathcal{U}$, we have

- $|\mathcal{U}| = mn_S^l$ poly log$B$, $|\mathcal{S}| = n_S^l$ poly log B
- $\Delta_S \leq B, f \leq$ poly log B for sufficient large B.

SC_B-CFM reduction. For each instance $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ of SC_B, we construct a graph $\mathcal{H} = (V, E)$ as follows (Fig. 6-2):

- Construct a bipartite graph with the vertex set $\mathcal{U} \cup \mathcal{S}$ and edges between S and all elements $e_i \in S$, for each $S \in \mathcal{S}$.
- Add a set D consisting of t vertices and a set D' with same number of vertices, say $D = \{x_1, x_2, \ldots, x_t\}$ and $D' = \{x'_1, x'_2, \ldots, x'_t\}$, where $t = \frac{|\mathcal{U}|}{B \ln^2 B}$.
- Connect x_i to $x'_i, \forall i = 1 \ldots t$. This enforces the selection of x_i in the optimal CFM.
- Connect each vertex $e_j \in \mathcal{U}$ to $\lceil \frac{\rho}{1-\rho} f(e_j) \rceil - 1$ and each vertex $S_k \in \mathcal{S}$ to $\lceil \frac{\rho}{1-\rho} |S_k| \rceil$ vertices in D, where $f(e_j)$ is the frequency of element e_j. During the connection, we balance the degrees of vertices in D.

We can assume w.l.o.g. that optimal solutions of CFM contains all vertices in D but not ones in D'. Then, all vertices in $\mathcal{S}$ will be activated after the first round, and

the a vertex in $\mathcal{U}$ is activated if and only if one of its neighbors in $\mathcal{S}$ is selected into the solution. Thus, the following lemma holds.

Lemma 19. *The size difference between the optimal CFM of $\mathcal{H}$ and the optimal SC_B of $\mathcal{F}$ is exactly the cardinality of D, i.e., $OPT_{CFM}(\mathcal{H}) = OPT_{SC}(\mathcal{F}) + t$.*

The key to preserve the hardness ratio is to keep the degree of vertices in $\mathcal{H}$ bounded and the gap between the optimal solutions' sizes small.

Lemma 20. *If $t = \frac{|\mathcal{U}|}{B \ln^2 B}$, then the maximum degree of vertices in $\mathcal{H}$ will be $B' = \Delta(\mathcal{H}) = O(B$ poly log $B)$.*

Proof. We can verify that vertices in $\mathcal{S}$ and $\mathcal{U}$ have degree $O(B)$. Vertices in D have degrees at most $\frac{\text{vol}(D)}{t} + 1$, where vol$(D)$ is the total degree of vertices in D. Define $\phi(X, Y)$ as the set of edges crossing between two vertex subsets X and Y. We have

$$\begin{aligned}
&\text{vol}(D) = |\phi(D, D')| + |\phi(D, \mathcal{U})| + |\phi(D, \mathcal{S})| \\
&= |D| + \sum_{S_k \in \mathcal{S}} \lceil \frac{\rho}{1-\rho} |S_k| \rceil + \sum_{e_j \in \mathcal{U}} \lceil \frac{\rho}{1-\rho} f(e_j) - 1 \rceil \\
&\leq \frac{2\rho}{1-\rho} |\mathcal{S}| B + |\mathcal{S}| + t = \left(\frac{2\rho}{1-\rho} B + 1 \right) |\mathcal{S}| + t
\end{aligned} \tag{6–4}$$

We have used the facts that $\sum_{S_k \in \mathcal{S}} |S_k| = \sum_{e_j \in \mathcal{U}} f(e_j)$ and $|S_k| \leq B,\ \forall S_k \in \mathcal{S}$.

Thus,

$$\begin{aligned}
B' &\leq \frac{1}{t} \left(\left(\frac{2\rho}{1-\rho} B + 1 \right) |\mathcal{S}| + t \right) + 1 \\
&\leq \left(\frac{2\rho}{1-\rho} B + 1 \right) \frac{B \ln^2 B \; n_S^l \text{ poly log } B}{m n^l \text{ poly log } B} \\
&\leq O(B \text{ poly log } B)
\end{aligned} \tag{6–5}$$

This completes the proof. □

Theorem 6.3. *When $d = 1$, it is NP-hard to approximate the CFM problem in graphs with degrees bounded by B' within a factor of $\ln B' - c_1 \ln \ln B'$, for some constant $c_1 > 0$.*

Proof. We prove by contradiction. Assume there exists algorithm $\mathcal{A}$ to find in graph with degrees bounded by B' and $d = 1$ a CFM of size at most $(\ln B' - c_1 \ln\ln B')\text{OPT}_{CFM}$, where OPT_{CFM} is the size of an optimal CFM. Let $\mathcal{F} = (\mathcal{U}, \mathcal{S})$ be an instance of SC_B with the optimal solution of size OPT_{SC}. Construct an instance $\mathcal{H}$ of CFM problem using the reduction SC_B-CFM as shown above. From (6–5), there exists constant $\beta > 0$ so that $B' \leq B \ln^{\beta} B$. Using algorithm $\mathcal{A}$ on $\mathcal{H}$, we obtain a solution of size at most $(\ln B' - c_1 \ln\ln B')\text{OPT}_{CFM}$. We can then convert that to a solution of SC_B by excluding vertices in D (see Lemma 19) and obtain a set cover of size at most

$$(\ln B' - c_1 \ln\ln B')(\text{OPT}_{SC} + t) - t \tag{6–6}$$

Since each set in $\mathcal{S}$ can cover at most B elements, we have $\text{OPT}_{SC} \geq \frac{|\mathcal{U}|}{B} = \frac{tB\ln^2 B}{B}$, thus $t \leq \frac{\text{OPT}_{SC}}{\ln^2 B}$. If we select $c_1 = c_0 + \beta + 1$, the solution of SC_B is then, after some algebra, at most $(\ln B - c_0 \ln\ln B)\text{OPT}_{SC}$ that contradicts the Lemma 18. □

Similarly, with appropriate setting in Feige's construction [38], we obtain the following hardness result regarding the network size n (the proof detail can be found in the technical report on our website).

Theorem 6.4. *For any $\epsilon > 0$, the CFM problem, when $d = 1$, cannot be approximated within a factor $(\frac{1}{2} - \epsilon)\ln n$, unless NP $\subset$ DTIME$(n^{O(\log\log n)})$.*

Proof. We use the same gadget in Fig. 6-2 to prove the hardness for CFM. Since, we no longer need to keep degree of vertices in the gadget bounded, we form a clique with vertices in D.

We can connect each $v \in (\mathcal{S} \cup \mathcal{U})$ to μ_v vertices in D. That is

$$|D| = O(\max_{v \in (\mathcal{S}\cup\mathcal{U})\mu_v} \theta(\frac{\rho}{1-\rho}\Delta_{\mathcal{S}})) = O(\Delta_{\mathcal{S}}) = O(m3^{l/2})$$

Or equivalently

$$|D|^2 = O(\sum_{v\in(\mathcal{S}\cup\mathcal{U})} d(v) + x_0(|\mathcal{S}| + |\mathcal{U}|) = O(2\sum_{v\in\mathcal{U}} d(v) + |\mathcal{S}| + |\mathcal{U}|) = O(m\,R\,k\,2^l)$$

To summarize, the sufficient condition is

$$|D| = O(m2^{\theta(l)} + (mRk2^l)^{1/2}). \tag{6–7}$$

By Lemma 17 and the construction, the hardness ratios of our problems are given by

$$\frac{(1-\frac{4}{k})kQ\ln m + |D|}{kQ + |D|}.$$

Unfortunately, with the same setting in the Feige's reduction, $|D| = O(\Delta_{\mathcal{S}}) = O((5n)^{\frac{2l}{\epsilon}}2^{\theta(l)})$, the above hardness ratio gets arbitrary close to 1. Hence we use a different setting in which $m = (5n)^{cl}$ with a small constant $c > 0$ to reduce the maximum degree. The consequence is that the inapproximability ratio is reduced accordingly.

The optimal setting to get the best inapproximability ratio is to set $m = (5n)^{l(1-\epsilon)}$ for some $\epsilon > 0$. Then, $N = mR = (5n)^{l(2-\epsilon)}$, or $m = N^{\frac{1-\epsilon}{2-\epsilon}}$. From (6–7), it is sufficient that

$$|D| = n^l\frac{2^{\theta(l)}}{n^{l\frac{\epsilon}{2}}} = o(Q)$$

Hence, the hardness ratio will be

$$\frac{(1-\frac{4}{k})kQ\ln m + o(Q)}{kQ + o(Q)} > (1-\frac{5}{k})\ln m$$

The number of vertices in the graph, denoted by $n_{\mathcal{H}}$, is

$$n_{\mathcal{H}} = 2|D| + |\mathcal{S}| + |\mathcal{U}| < \theta(m3^{l/2}) + n^l 2^{2l}\left(\frac{5}{3}\right)^{l/2} + (5n)^{2l-\epsilon} < 2|\mathcal{U}| = 2N$$

Finally, the hardness ratio is at least

$$(1-\frac{5}{k})\ln\left(\frac{n_{\mathcal{H}}}{2}\right)^{1/2-\frac{\epsilon}{4-2\epsilon}} > (1-\frac{5}{k})\frac{1}{2}\left(1-\frac{\epsilon}{2-\epsilon}\right)\ln n_{\mathcal{H}} - \theta(1) > \frac{1}{2}(1-\epsilon)\ln n_{\mathcal{H}}.$$

Here, we assume k is sufficiently large and ϵ is sufficiently small. □ □

Note that Theorems 6.3 and 6.4 are incomparable in general. Let Δ be the maximum degree, Theorem 6.3 implies the hardness of approximation with factor $(1-\epsilon)\ln\Delta$, which is larger than $(\frac{1}{2}-n)\ln n$ if $\Delta \approx n$, but smaller when $\Delta < \sqrt{n}$, for example in power-law graphs with the exponent $\gamma > 2$. In addition, the Theorem 6.4 uses a stronger assumption than that in Theorem 6.3.

6.3.3 Multiple-hop CFM

We now present a gap reduction from the CFM problem to the one-hop CFM problem with $d \geq 2$. The hardness result follows immediately by the Theorem 6.3 in the previous section.

Given a graph $G = (V, E)$ as an instance of the CFM problem. We will construct an instance $G' = (V', E')$ of the CFM problem as follows (and as illustrated in Fig. 6-4). We

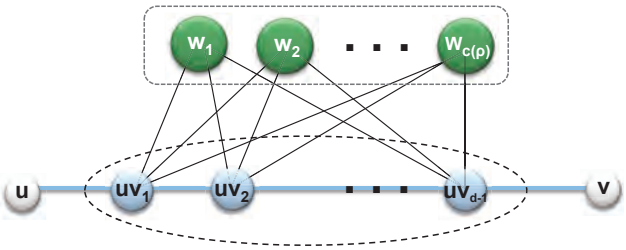


Figure 6-3. The transmitter gadget.

add $c(\rho)$ vertices $w_1, w_2, \ldots, w_{c(\rho)}$, called flashpoints, where $c(\rho) = \min\{t \in \mathbb{N} \,|\frac{t-1}{t+1} \leq \rho < \frac{t}{t+1}\}$. These vertices will be selected at the beginning to kick off the activation of other nodes. Furthermore, each "flashpoint" w_p is connected to a dummy vertex z_p.

Replace each edge $(u, v) \in E$ by a gadget called transmitter. The transmitter connecting vertex u and v is a chain of $d-1$ path, named uv_1 to uv_{d-1}. The vertex u is connected to uv_1, uv_1 is connected to uv_2 and so on, vertex uv_{d-1} is connected to v. Each vertex uv_i, $i = 1..d-1$ is connected to all flashpoints. An example for transmitter is shown in Fig. 6-3. The transmitter is designed so that if all flashpoints and vertex u are selected at the beginning, then vertex uv_{d-1} will be activated after $d-1$ rounds.

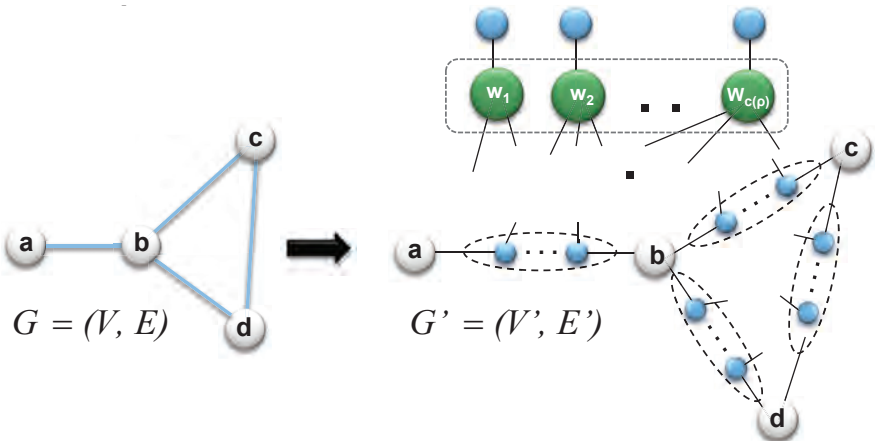


Figure 6-4. Gap-reduction from one-hop CFM to d-hop CFM.

Hence, the number of activated neighbors of v after $d-1$ rounds will equal the number of selected neighbors of v in the original graph.

Finally, we replace each edge (w_p, z_p) by a transmitter. In order to activate all dummy vertices z_p after d rounds, we can assume, w.l.o.g., that all flashpoints must be selected in an optimal solution. The following lemma follows directly from the construction.

Lemma 21. *Every solution of size k for the one-hop ($d = 1$) CFM problem in G induces a solution of size $k + c(\rho)$ for the d-hop CFM problem in G'.*

On another direction, we also have the following lemma.

Lemma 22. *An optimal solution of size k' for the d-hop CFM problem induces a size $k' - c(\rho)$ solution for the one-hop CFM problem in G.*

Proof. For a transmitter connecting u to v, if the solution of the d-hop CFM problem contains any of the intermediate vertices $uv_1, \ldots, uv_{d-1}$, we can replace that vertex in the solution with either u or v to obtain a new solution of same size (or less). Hence, we can assume, w.l.o.g., that none of the intermediate vertices are selected. Therefore, all flashpoints must be selected in order to activate the dummy vertices. It is easy to see that the solution of d-hop CFM excluding the flashpoints will be a solution of one-hop CFM in G with size $k' - c(\rho)$. □

Note that the number of vertices in G' is upper-bounded by dn^2 i.e. $\ln|V'| < 2ln|V| + lnd$. Thus, using the same arguments used in the proof of Theorem 6.4, we can show that a $(\frac{1}{4} - \epsilon)\ln n$ approximation algorithm algorithm lead to a $(\frac{1}{2} - \epsilon)\ln n$ approximation algorithm for the one-hop CFM problem (contradicts Theorem 6.4).

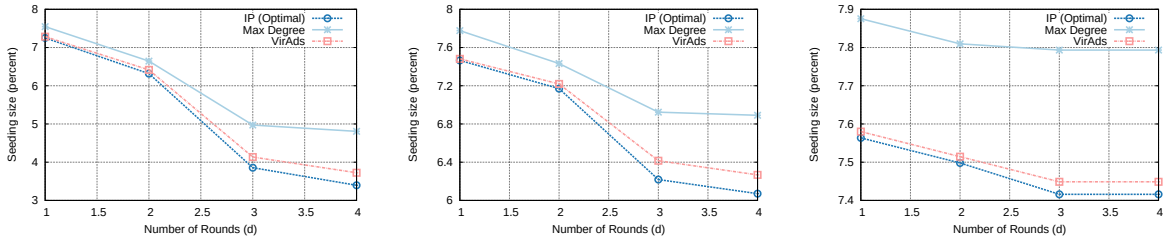
Theorem 6.5. *The CFM problem cannot be approximated within* $(\frac{1}{4} - \epsilon)\log n$ *for* $d \geq 1$*, unless NP* $\subset$ *DTIME*$(n^{O(\log\log n)})$

6.4 Empirical Study

In this section we perform experiments on OSNs to show the efficiency of our algorithms in comparison with simple degree centrality heuristic and study the trade-off between the number of times the information is allowed to propagate in the network and the seeding size.

6.4.1 Comparing to Optimal Seeding

One advantage of our discrete diffusion model over probabilistic ones [49, 50] is that the exact solution can be found using mathematical programming. This enables us to study the exact behavior of the seeding size when the number of propagation hop varies.



A $\rho = 0.4$ B $\rho = 0.6$ C $\rho = 0.8$

Figure 6-5. Seeding size (in percent) on Erdos's Collaboration network. VirAds produces close to the optimal seeding in only fractions of a second (in comparison to 2 days running time of the IP(optimal))

We formulate the CFM problem as an $0-1$ Integer Linear Programming (ILP) problem below.

$$\text{minimize} \sum_{v \in V} x_v^0 \tag{6–8}$$

$$\text{subject to} \sum_{v \in V} x_v^d \geq |V| \tag{6–9}$$

$$\sum_{w \in N(v)} x_w^{i-1} + \lceil \rho \cdot d(v) \rceil x_v^{i-1} \geq \lceil \rho \cdot d(v) \rceil \; x_v^i$$

$$\forall v \in V, i = 1..d \tag{6–10}$$

$$x_v^i \geq x_v^{i-1} \qquad \forall v \in V, i = 1..d \tag{6–11}$$

$$x_v^i \in \{0, 1\} \qquad \forall v \in V, i = 0..d \tag{6–12}$$

where $x_v^i = \begin{cases} 0 & \text{if v is inactive at round } i \\ 1 & \text{otherwise} \end{cases}$.

The objective of the ILP is to select a minimum number of seeds at the beginning. The constraint (2) guarantees all nodes are activated at the end, while (3) deals with propagation condition; the constraint (4) is simply to keep vertices active once they are activated.

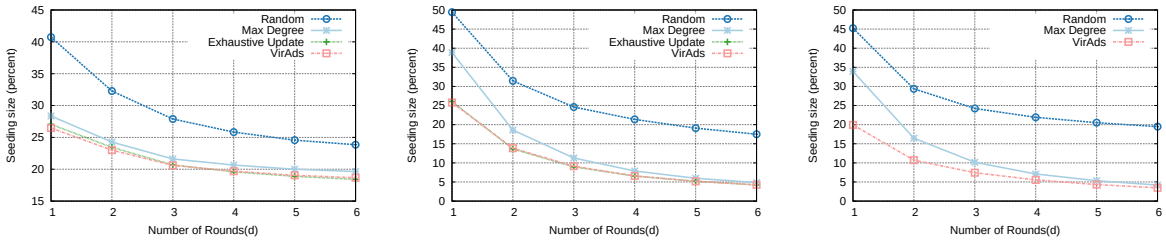
We solve the ILP problem on Erdos collaboration networks, the social network of famous mathematician, [13]. The network consists of 6100 vertices and 15030 edges. The ILP is solved with the optimization package GUROBI 4.5 on Intel Xeon 2.93 Ghz PC and setting the time limit for the solver to be 2 days. The running time of the IP solver increases significantly when d increases. For $d = 1, 2,$ and 3, the solver return the optimal solutions. However, for $d = 4$, the solver cannot find the optimal solutions within the time limit and returns sub-optimal solutions with relative errors at most 15%.

The optimal (or sub-optimal) seeding sizes are shown in Figs. 6-5A, 6-5B, and 6-5C for $\rho = 0.4, 0.6$ and 0.8, respectively. VirAds provides close-to-optimal solutions and performs much better Max Degree. Especially, when $\rho = 0.8$ the VirAds's seeding is only

different with the optimal solutions by one or two nodes. In addition, VirAds only takes fractions of a second to generate the solutions.

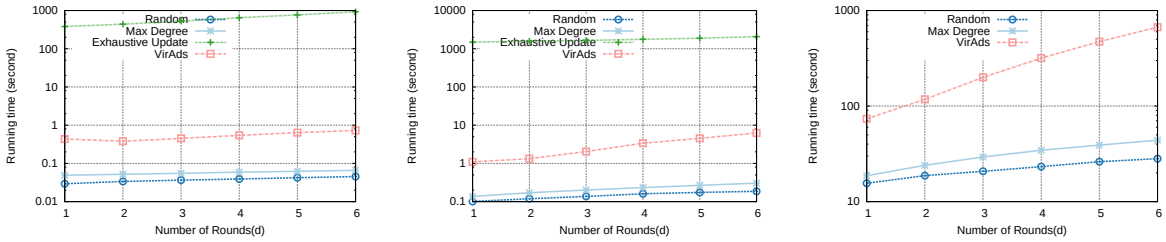
As proven in Section 6.1, the seeding takes a constant fraction of nodes in the network. For Erdos Colloboration Network, the seeding consists of 3.8% to 7% the number of nodes in the networks. Further, the seeding can consist as high as 20% to 40% nodes in the network for larger social networks in next section.

Although the mathematical approach can provide accurate measurement on the optimal seeding size, it cannot be applied for larger networks. The rest of our experiments measures the quality and scalability of our proposed algorithm VirAds on a collection of large networks.



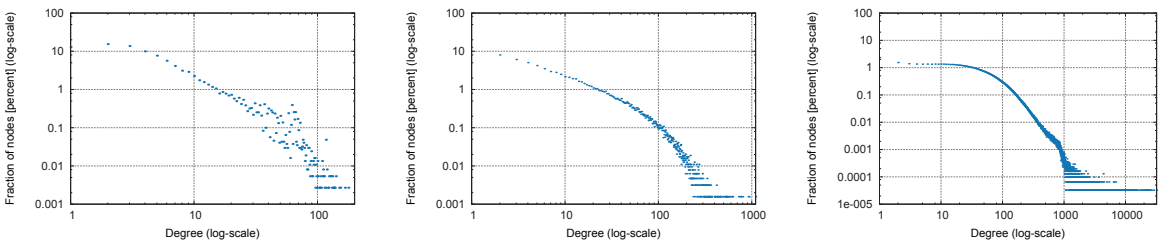
A Physics B Facebook C Orkut

Figure 6-6. Seeding size when the number of propagation hop d varies ($\rho = 0.3$). VirAds consistently has the best performance.



A Physics B Facebook C Orkut

Figure 6-7. Running time when the number of propagation hop d varies ($\rho = 0.3$). Even for the largest network of 110 million edges, VirAds takes less than 12 minutes.



A Physics B Facebook C Orkut

Figure 6-8. Degree distribution of studied networks

6.4.2 Large Social Networks

We select networks of various sizes including Coauthors network in Physics sections of the e-print arXiv [49], Facebook [76] and Orkut [59], a social networking run by Google. Links in all three networks are undirected and unweighted. The sizes of the networks are presented in Table 6-1. The degree distributions of those networks are shown in Fig. 6-8.

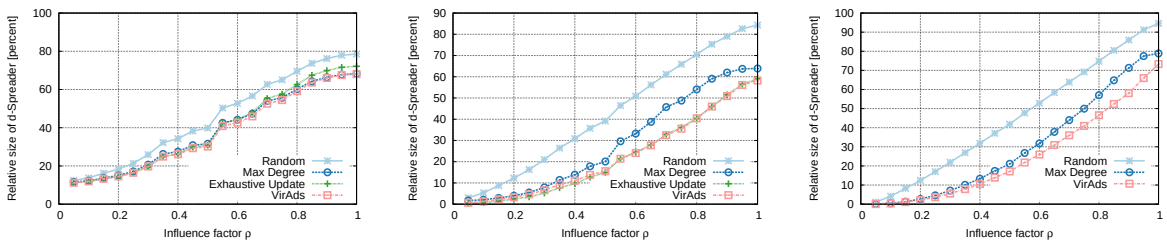
Table 6-1. Sizes of the investigated networks

	Physics	**Facebook**	**Orkut**
Vertices	37,154	90,269	3,072,441
Edges	231,584	3,646,662	223,534,301
Avg. Degree	12.5	80.8	145.5

Physics: We shall refer the physics coauthors network as Physics network or simply Physics. Each node in the network represents an author and there is an edge between two authors if they coauthor one or more papers. *Facebook* dataset consists 52% of the users in the New Orleans [76]. *Orkut* dataset is collected by performing crawling in last 2006 [59]. It contains about 11.3% of Orkut's users.

6.4.3 Solution Quality in Large Social Networks

We compare our VirAds algorithm with the following heuristics *Random* method in which vertices are picked up randomly until forming a d-seeding and *Max Degree* method in which vertices with highest degree are selected until forming a d-hop seeding.



A Physics B Facebook C Orkut

Figure 6-9. Seeding size at different influence factors ρ (the maximum number of propagation hops is $d = 4$).

Finally, we compare VirAds with its naive implementation, called *Exhaustive Update*, in which after selecting a vertex into the seeding, the effectiveness of all the remaining vertices are recalculated. With more accurate estimation on vertex effectiveness, Exhaustive Search is expected to produce higher quality solutions than those of VirAds.

The seeding size with different number of propagation hop d when $\rho = 0.3$ are shown in Fig. 6-6. To our surprise, VirAds even performs equal or better than *Exhaustive Update* despite that it uses significantly less effort to update vertex effectiveness. VirAds has smaller seeding in Physics than *Exhaustive Update*; both of them give similar results for Faceboook; while *Exhaustive Update* cannot finish on Orkut after 48 hours and was forced to terminate. Sparingly update the vertices' effectiveness turns out to be efficient enough since the influence propagation is locally bounded. In addition, the seeds produced by VirAds are almost two times smaller than those of *Random*.

The gap between VirAds and Max Degree is narrowed when the number of maximum hops increases. Hence, selecting nodes with high degrees as seeding is a good long-term strategy, but might not be efficient for fast propagation when the number of hops is limited. In Facebook and Orkut, when $d = 1$, *Max Degree* has 60% to 70% more vertices in the seeding than *VirAds*. In Physics, the gap between VirAds and the *Max Degree* is less impressive. Nevertheless, VirAds consistently produces the best solutions in all networks.

6.4.4 Scalability

The running time of all methods at different propagation hop d are presented in Fig 6-7. The time is measured in second and presented in the log scale. The running times increase slightly together with the number of propagation rounds d, and are proportional to the size of the network. The *Exhaustive Update* has the worst running time, taking up to 15 minutes for Physics, 20 minutes for Facebook. For Orkut, the algorithm cannot finish within 2 days, as mentioned. The three remaining algorithms *VirAds*, *Max Degree*, and *Random* take less than one second for Physics, and less than 10 seconds for Facebook. Even on the largest network Orkut with more than 220 million edges, VirAds requires less than 12 minutes to complete.

6.4.5 Influence factor

We study the performance of VirAds and the other method at different influence factor ρ. The number of propagation rounds d is fixed to 4. The size of d-seeding sets are shown in Figures 6-9. VirAds is clearly still the best performer. The seeding sizes of VirAds are up to 5 times smaller than those of Max Degree for small ρ (although it's hard to see this on the charts due to small seeding sizes).

Since all tested networks are social networks with small diameter, the seeding sizes go to zero when ρ is close to zero. The exception is the Physics, in which the seeding sizes do not go below 10% the number of vertices in the networks even when $\rho = 0.05$. A closer look into the Physics network reveals that the network contain many isolated cliques of small sizes (2, 3, 4, and so on) which correspond to authors that appear in only one paper. In each clique, regardless of the threshold ρ, at least one vertex must be selected, thus the seeding size cannot get below the number of isolated cliques in the networks. To eliminate the effect of isolated cliques, a possible approach is to restrict the problem to the largest component in the network.

CHAPTER 7
CONCLUSION

Society relies heavily on its networked physical infrastructure and information systems. To detect vulnerability issues in a network, it is of particular importance to analyze how well-connected the network will remain after a disruptive event takes place. We propose the use of *pairwise connectivity*, the number of connected pairs in the network, as a disruptive effect measurement, and use it to formulate network vulnerability assessment as optimization problems. The objective is to identify the minimum set of *critical network elements* (nodes or edges) whose removal results in a major degradation of the network pairwise connectivity.

We prove that both critical *edges* detection (CED) and critical *nodes* detection (CND) are NP-complete [33]; and develop two novel solutions with provable guarantees: 1) an $O(\log^{1.5} n)$ bicriteria approximation algorithm for CED based on constructing a decomposition tree with recursive c-balanced cut and 2) an $O(\log n \log\log n)$ bicriteria approximation algorithm for CND [31]. Later we design a bicriteria approximation algorithm with performance guarantee $O(\sqrt{\log n})$ when the set of critical elements may include both edges and nodes. This immediately implies improved results for both CED and CND. The extensive experiments have revealed many insights on the relative criticality between edges and nodes in the networks on different network topologies.

dynamic networks, e.g. cellular networks, or mobile sensor networks, detecting critical nodes is extremely challenging due to the continual changes in network topology. We abstract dynamic networks as probabilistic graphs and measure the disruptive effect in terms of *expected pairwise connectivity* (EPC). Computing EPC is tightly related to network reliability problems, some of the most classical open #P-complete problems. Beyond showing #P-completeness of EPC, we have approximated EPC with an *FPRAS*, which gives a potential direction to tackle open questions in network reliability. Further, we formulate the problem of detecting critical nodes as a two-level

stochastic programming and present a sample average approximation algorithm to solve the formulation with guaranteed accuracy.

We investigate in Chapter 6 cascading failures in complex systems. Those failures often propagate and lead to a much more devastating consequence. Thus, it is crucial to detect critical nodes whose failures will trigger a cascading failure to an entire network, leaving major nodes in the failure state within a given number of steps. My theoretical analysis shows that *the cascading of failures maybe quite different in power-law networks* than others. First, we prove that a large number of initial failures are required to trigger a network-wide failure. Second, the problem of detecting critical nodes cannot be approximated within a factor $O(\log n)$ in general graphs, however, there is a constant factor approximation algorithm for the problem in power-law networks. Extensive experiments on large-scale OSNs up to hundreds of millions of edges demonstrate the effectiveness of my proposed algorithm. My study is also applied naturally to the problems of information propagation, viral marketing, and disease spreading.

REFERENCES

[1] "US IP Backbone network XO company." urlhttp://www.xo.com/about/network/Pages/overview.aspx, ????

[2] Agarwal, A., Charikar, M., Makarychev, K., and Makarychev, Y. "O(log n) approximation algorithms for min UnCut, min 2CNF deletion, and directed cut problems." *STOC*. New York, NY, USA: ACM, 2005, 573–581.

[3] Aiello, W., Chung, F., and Lu, L. "A random graph model for massive graphs." *STOC '00*. New York, NY, USA: ACM, 2000.

[4] ———. "A Random Graph Model for Power Law Graphs." *Experimental Math* 10 (2000): 53–66.

[5] Aiello, William, Chung, Fan, and Lu, Linyuan. "Random Evolution in Massive Graphs." *In Handbook of Massive Data Sets*. Kluwer Academic Publishers, 2001.

[6] Albert, R., Albert, I., and Nakarado, G. L. "Structural Vulnerability of the North American Power Grid." *Phys. Rev. E* 69 (2004).2: 10.

[7] Albert, R., Jeong, H., and Barabasi, A. "Error and attack tolerance of complex networks." *Nature* 406 (2000).6794: 378–382.

URL http://dx.doi.org/10.1038/35019019

[8] Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

URL http://books.google.com/books?id=nGvI7cOuOOQC

[9] Arora, S., Hazan, E., and Kale, S. "O($\sqrt{\log n}$) Approximation to SPARSEST CUT in $\tilde{O}(n^2)$ Time." *SIAM J. Comput.* 39 (2010).5.

[10] Arulselvan, A., Commander, Clayton W., Elefteriadou, L., and Pardalos, Panos M. "Detecting critical nodes in sparse graphs." *Computers & Operations Research* 36 (2009).7: 2193–2200.

[11] ———. "Detecting critical nodes in sparse graphs." *Computers and Operations Research* 36 (2009).7.

[12] Barabasi, A., Albert, R., and Jeong, H. "Scale-free characteristics of random networks: the topology of the world-wide web." *Physica A* 281 (2000).

[13] Barabasi, A, Jeong, H, Neda, Z, Ravasz, E, Schubert, A, and Vicsek, T. "Evolution of the social network of scientific collaborations." *Physica A: Statistical Mechanics and its Applications* 311 (2002).3-4: 590–614.

URL http://linkinghub.elsevier.com/retrieve/pii/S0378437102007367

[14] Bissias, G. D. *Bounds on service quality for networks subject to augmentation and attack*. Ph.D. thesis, University of Massachusett Amherst, 2010.

[15] Bissias, George, Levine, Brian Neil, and Rosenberg, Arnold L. "Bounding Damage From Link Destruction with Application to the Internet (extended abstract)." *Proc. ACM SIGMETRICS*. 2007, 367—368.

URL http://prisms.cs.umass.edu/brian/pubs/bissias.sigmetrics.abstract.2007.pdf

[16] Blackford, L. S., Choi, J., Cleary, A., D'Azeuedo, E., Demmel, J., Dhillon, I., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., and Whaley, R. C. *ScaLAPACK user's guide*. SIAM, 1997.

[17] Borgatti, Stephen P. "Identifying sets of key players in a social network." *Computational & Mathematical Organization Theory* 12 (2006).1: 21–34.

[18] Borgatti, Stephen P. and Everett, Martin G. "A Graph-theoretic perspective on centrality." *Social Networks* 28 (2006).4: 466–484.

URL http://dx.doi.org/10.1016/j.socnet.2005.11.005

[19] Boyd, S.P. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

[20] Cauchy, A.L.B. and polytechnique (France), École. *Cours d'analyse de l'École royale polytechnique*. No. v. 1 in Cours d'analyse de l'École royale polytechnique. Imprimerie royale, 1821.

URL http://books.google.com/books?id=n60AAAAAMAAJ

[21] Centola, Damon and Macy, Michael. "Complex Contagions and the Weakness of Long Ties." *American Journal of Sociology* 113 (2007).3: 702–734.

URL http://dx.doi.org/10.1086/521848

[22] Chen, N. "On the Approximability of Influence in Social Networks." *SIAM Journal of Discrete Mathematics* 23 (2009).3: 1400–1415.

[23] Chung, Fan R. K. *Spectral Graph Theory (CBMS Regional Conference Series in Mathematics, No. 92)*. American Mathematical Society, 1997.

[24] Church, R., Scaparra, M., and Middleton, R. "Identifying critical infrastructure: the median and covering facility interdiction problems." *Ann Assoc Am Geogr* 94 (2004).3.

[25] Clauset, A., Shalizi, C. R., and Newman, M. E. J. "Power-law distributions in empirical data." *SIAM Reviews* (2007).

[26] Colbourn, Charles J. *The Combinatorics of Network Reliability*. New York, NY, USA: Oxford University Press, Inc., 1987.

[27] Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. *Introduction to Algorithms*. The MIT Press, 2009, 3rd edition ed.

[28] Dagum, P., Karp, R., Luby, M., and Ross, S. "An Optimal Algorithm for Monte Carlo Estimation." *SIAM Journal on Computing* 29 (2000).5: 1484–1496.

URL http://epubs.siam.org/doi/abs/10.1137/S0097539797315306

[29] Demmel, J. W., Eisenstat, S. C., Gilbert, J. R., Li, X. S., and Liu, J. W. H. "A supernodal approach to sparse partial pivoting." *SIAM J. Matrix Analysis and Applications* 20 (1999).3: 720–755.

[30] Dinh, T. N. and Thai, M. T. "Precise Structural Vulnerability Assessment Via Mathematical Programming." *Proc. of IEEE MILCOM*. 2011.

[31] ———. "Network under Joint Node and Link Attacks: Vulnerability Assessment Methods and Analysis." Tech. rep., Dept. of CISE, University of Florida, 2012.

[32] Dinh, T. N., X., Ying, Thai, M. T., Park, E.K., and Znati, T. "On Approximation of New Optimization Methods for Assessing Network Vulnerability." *Proc. of IEEE INFOCOM*. 2010.

[33] Dinh, Thang N., Xuan, Ying, Thai, My T., Park, E. K., and Znati, Taieb. "On approximation of new optimization methods for assessing network vulnerability." *INFOCOM*. Piscataway, NJ, USA: IEEE Press, 2010, 2678–2686.

[34] Dinur, I. and Safra, S. "On the Hardness of Approximating Minimum Vertex Cover." *Annals of Mathematics* 162 (2004): 2005.

[35] Donath, W. E. and Hoffman, A. J. "Lower bounds for the partitioning of graphs." *IBM J. Res. Dev.* 17 (1973).

[36] Erdos, P. and Renyi, A. "On the evolution of random graphs." *Publ. Math. Inst. Hungary. Acad. Sci.* 5 (1960): 17–61.

[37] Even, G., Naor, J. S., Rao, S., and Schieber, B. "Divide-and-conquer approximation algorithms via spreading metrics." *J. of ACM* 47 (2000).4: 585–616.

[38] Feige, U. "A threshold of ln n for approximating set cover." *Journal of ACM* 45 (1998).4: 634–652.

[39] Ferrante, Alessandro. "Hardness and Approximation Algorithms of Some Graph Problems." 2006.

[40] Garey, Michael R. and Johnson, David S. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[41] Gkantsidis, C., Mihail, M., and Saberi, A. "Conductance and congestion in power law graphs." *SIGMETRICS '03: Proceedings of the International Conference on Measurements and Modeling of Computer Systems*. New York, NY, USA: ACM, 2003, 148–159.

[42] Goldberg, A V and Tarjan, R E. "A new approach to the maximum flow problem." *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. STOC '86. New York, NY, USA: ACM, 1986, 136–146.

URL http://doi.acm.org/10.1145/12130.12144

[43] Goyal, A., Bonchi, F., and Lakshmanan, L. V. S. "Learning influence probabilities in social networks." *WSDM '10* (2010): 241–250.

URL http://portal.acm.org/citation.cfm?id=1718518

[44] Goyal, D. and Caffery, J. "Partitioning avoidance in mobile Ad Hoc networks using network survivability concepts." *ISCC* (2002): 553.

[45] Grtschel, M. and Wakabayashi, Y. "A cutting plane algorithm for a clustering problem." *Mathematical Programming* 45 (1989). 10.1007/BF01589097.

[46] Grubesic, Tony H., Matisziw, Timothy C., Murray, Alan T., and Snediker, Diane. "Comparative Approaches for Assessing Network Vulnerability." *Inter. Regional Sci. Review* 31 (2008).

URL http://dx.doi.org/10.1177/0160017607308679

[47] Hauspie, M., Carle, J., and Simplot, D. "Partition detection in mobile Ad Hoc networks using multiple disjoint paths set." *Workshop of Objects, Models and Multimedia technology* (2003).

[48] Jorgic, M., Stojmenovic, I., Hauspie, M., and Simplot-Ryl, D. "Localized algorithms for detection of critical nodes and links for connectivity in ad hoc networks." *3rd IFIP MED-HOC-NET Workshop* (2004).

[49] Kempe, D., Kleinberg, J., and Tardos, É. "Maximizing the spread of influence through a social network." *KDD'03: Proceedings of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2003, 137–146.

[50] Kempe, D., Kleinberg, J., and Tardos, E. "Influential nodes in a diffusion model for social networks." *International Colloquium on Automata, Languages and Programming '05*. 2005, 1127–1138.

[51] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. "Optimization by Simulated Annealing." *Science* 220 (1983).4598: 671–680.

[52] Kleywegt, A., Shapiro, A., and Homem-de Mello, T. "The Sample Average Approximation Method for Stochastic Discrete Optimization." *SIAM Journal on Optimization* 12 (2002).2: 479–502.

[53] Lehman, T., Sobieski, J., and Jabbari, B. "DRAGON: a framework for service provisioning in heterogeneous grid networks." *IEEE Communication Magazines* (2006).

[54] Lehoucq, R. B., Sorensen, D. C., and Yang, C. "ARPACK Users Guide: Solution of Large Scale Eigenvalue Problems by Implicitly Restarted Arnoldi Methods." 1997.

[55] Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., and Glance, N. "Cost-effective outbreak detection in networks." *ACM SIGKDD Conference on Knowledge Discovery and Data Mining '07*. New York, NY, USA: ACM, 2007, 420–429.

[56] Leskovec, Jure, Kleinberg, Jon, and Faloutsos, Christos. "Graphs over time: densification laws, shrinking diameters and possible explanations." *KDD*. ACM, 2005, 177–187.

[57] Meila, M. and Pentney, W. "Clustering by Weighted Cuts in Directed Graphs." *Proceedings of the SIAM Conference on Data Mining*. 2007.

[58] Mhatre, V. and Rosenberg, C. "Homogeneous vs heterogeneous clustered sensor networks: a comparative study." *IEEE ICC* (2004).

[59] Mislove, A., Marcon, M., Gummadi, Krishna P., Druschel, P., and Bhattacharjee, B. "Measurement and Analysis of Online Social Networks." *IMC'07*. San Diego, CA, 2007.

[60] Mladenovic, N. and Hansen, P. "Variable neighborhood search." *Computers & Operations Research* 24 (1997).11: 1097 – 1100.

[61] Mohar and Poljak. "Eigenvalue in combinatorial optimization." *Combinatorial and Graph-Theoretical Problems in Linear Algebra* (1992).

[62] Murray, A., Matisziw, T., and Grubesic, T. "Multimethodological approaches to network vulnerability analysis." *Growth Change* (2008).

[63] Neumayer, S., Zussman, G., Cohen, R., and Modiano, E. "Assessing the Vulnerability of the Fiber Infrastructure to Disasters." *Proc. of IEEE INFOCOM*. 2009.

[64] Neumayer, Sebastian, Zussman, Gil, Cohen, Reuven, and Modiano, Eytan. "Assessing the Vulnerability of the Fiber Infrastructure to Disasters." *IEEE/ACM Trans. Netw.* (2011): 1610–1623.

[65] Page, L., Brin, S., Motwani, R., and Winograd, T. "The PageRank Citation Ranking: Bringing Order to the Web." Tech. rep., Stanford InfoLab, 1999.

[66] Peleg, D. “Local Majority Voting, Small Coalitions and Controlling Monopolies in Graphs: A Review.” *SIROCCO’96: Colloquium on Structural Information and Communication Complexity*. 1996, 152–169.

[67] Pinar, A., Meza, J., Donde, V., and Lesieutre, B. “Optimization Strategies for the Vulnerability Analysis of the Electric Power Grid.” *SIAM J. on Optimization* 20 (2010).

[68] Sen, A., Murthy, S., and Banerjee, S. “Region-based connectivity - a new paradigm for design of fault-tolerant networks.” *HPSR*. 2009.

[69] Shapiro, A., Dentcheva, D., and Ruszczyński, A.P. *Lectures on Stochastic Programming: Modeling and Theory*. MPS-SIAM Series on Optimization Series. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009.

[70] Sinclair, A. and Jerrum, M. “Approximate counting, uniform generation and rapidly mixing Markov chains.” *Inf. Comput.* 82 (1989).1: 93–133.

URL http://dx.doi.org/10.1016/0890-5401(89)90067-9

[71] Stoer, M. and Wagner, F. “A simple min-cut algorithm.” *J. of ACM* 44 (1997).4: 585–591.

[72] Suh, Y. J., Kim, D. J., Lim, W. S., and Baek, J. Y. “Method for supporting quality of service in heterogeneous networks.” 2009.

[73] Sun, Fangting and Shayman, Mark A. “On pairwise connectivity of wireless multihop networks.” *International Journal of Security and Networks* 2 (2007).1/2: 37–49.

[74] Trevisan, L. “Non-approximability results for optimization problems on bounded degree instances.” *ACM Symposium on Theory of Computing ’01*. New York, NY, USA: ACM, 2001, 453–461.

[75] Valiant, L. “The Complexity of Enumeration and Reliability Problems.” *SIAM Journal on Computing* 8 (1979).3: 410–421.

URL http://epubs.siam.org/doi/abs/10.1137/0208032

[76] Viswanath, B., Mislove, A., Cha, M., and Gummadi, K. P. “On the Evolution of User Interaction in Facebook.” *WOSN’09*. 2009.

[77] Wagner, D. and Wagner, F. “Between Min Cut and Graph Bisection.” *MFCS*. London, UK: Springer-Verlag, 1993, 744–750.

[78] Wang, F., Camacho, E., and Xu, K. "Positive Influence Dominating Set in Online Social Networks." *Proceedings of the 3rd International Conference on Combinatorial Optimization and Applications*. COCOA '09. Berlin, Heidelberg: Springer-Verlag, 2009, 313–321.

URL http://dx.doi.org/10.1007/978-3-642-02026-1_29

[79] Watts, D. J. and Strogatz, S. H. "Collective dynamics of 'small-world' networks." *Nature* 393 (1998).6684: 440–442.

URL http://dx.doi.org/10.1038/30918

[80] White, S. and Smyth, P. "A Spectral Clustering Approach To Finding Communities in Graph." *SDM*. 2005.

[81] Woo, Gordon. "Intelligence Constraints on Terrorist Network Plots." *Mathematical Methods in Counterterrorism*. eds. Nasrullah Memon, Jonathan David Farley, David L. Hicks, and Torben Rosenorn. Springer Vienna, 2009. 205–214.

URL http://dx.doi.org/10.1007/978-3-211-09442-6_12

[82] Z., Feng, Z., Zhao, and W., Weili. "Latency-Bounded Minimum Influential Node Selection in Social Networks." *Wireless Algorithms, Systems, and Applications*. eds. Benyuan Liu, Azer Bestavros, Ding-Zhu Du, and Jie Wang, Lecture Notes in Computer Science. 2009, 519–526.

URL http://dx.doi.org/10.1007/978-3-642-03417-6_51

[83] Zhu, X., Yu, J., Lee, W., Kim, D., Shan, S., and Du, D.-Z. "New dominating sets in social networks." *Journal of Global Optimization* 48 (2010): 633–642. 10.1007/s10898-009-9511-2.

URL http://dx.doi.org/10.1007/s10898-009-9511-2

BIOGRAPHICAL SKETCH

Thang N. Dinh received the BA degree in Information Technology from Vietnam National University, Hanoi, Vietnam in 2007. His research focuses on designing combinatorial optimization methods for dynamic complex networks and mobile adhoc network including network vulnerability, dynamic community structure, and fast information propagation.

THE EXPLOITATION OF POWER-LAW NETWORKS: ROBUSTNESS, OPTIMIZATION AND ITS IMPACT ON COMMUNICATION NETWORKS AND SOCIAL BEHAVIORS

By

YILIN SHEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

I dedicate this to my parents and my girlfriend.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my committee chair Dr. Thai for her priceless help for my Ph.D. program. Not only the chance that she offered me to further my study and research on Computer Networking for my Ph.D. degree, but also her precious guidance over my 4-year Ph.D. study and research, are indispensable to this thesis. Her strong passion, preciseness and profound knowledge for research have been enlightening me throughout my Ph.D. period. The financial support from her evades me from the financial problems for international students and so that I can concentrate over the researches.

I would also like to thank all the professors, Prof. Sartaj Sahni, Prof. Sanjay Ranka, Prof. Prabhat Mishra and Prof. Panos M. Pardalos, in my committee for their time for discussing over my research topics and providing numerous constructive opinions.

I would like to thank my group members, Ying Xuan, Thang N. Dinh, Nam P. Nguyen, Dzung T. Nguyen, Ravi Tiwari, Incheol Shin, Huiyuan Zhang for their help in my study and work.

My research was partially funded by DTRA, Young Investigator Award, Basic Research Program # HDTRA1-09-1-0061 and DTRA # HDTRA1-08-10.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

THE EXPLOITATION OF POWER-LAW NETWORKS: ROBUSTNESS, OPTIMIZATION AND ITS IMPACT ON COMMUNICATION NETWORKS AND SOCIAL BEHAVIORS

By

Yilin Shen

May 2013

Chair: My T. Thai
Major: Computer Engineering

Many practical complex networks, such as the Internet, WWW and social networks, are discovered to follow power-law distribution in their degree sequences, i.e., the number of nodes with degree i in these networks is proportional to $i^{-\beta}$ for some exponential factor $\beta > 1$. The exploitation of such networks becomes an urgent need, yet remains open especially from theoretical viewpoints.

In this dissertation, we first investigate if it is easier to solve many optimization problems in power-law networks. Our works focus on the hardness and inapproximability of optimization problems on power-law graphs (PLG). Particularly, we show that the MINIMUM DOMINATING SET, MINIMUM VERTEX COVER and MAXIMUM INDEPENDENT SET are still *APX*-hard on power-law graphs. We further show the inapproximability factors of these optimization problems and a more general problem (ρ-MINIMUM DOMINATING SET), which proved that a belief of $(1 + o(1))$-approximation algorithm for these problems on power-law graphs is not always true. In order to show the above theoretical results, we propose a general cycle-based embedding technique to embed any d-bounded graphs into a power-law graph. In addition, we present a brief description of the relationship between the exponential factor β and constant greedy approximation algorithms. Moreover, we propose a algorithm framework, called Low-Degree Percolation (LDP) Algorithm Framework, for solving Minimum Dominating Set, Minimum Vertex Cover and Maximum Independent Set problems in power-law

graphs. Using this framework, we further show a theoretical framework to derive the approximation ratios for these optimization problems in two well-known random power-law graphs. Numerical analysis shows that our proposed framework can not only lead to a good theoretical approximation ratio but also result in even better performance than theoretical bounds.

In addition, the robustness of power-law networks attracts more research attentions since they are exposed to a great number of threats such as adversarial attacks on the Internet, cybercrimes on the WWW or malware propagations on social networks. In this dissertation, we first show it NP-hard to detect critical links and nodes even in power-law networks. Due to the denial of promptly assessing vulnerability of power-law networks in this manner, we are more interested in the vulnerability of power-law networks under random attacks and adversarial attacks using the in-depth probabilistic analysis on the theory of random power-law graph models. Our results indicate that power-law networks are able to tolerate random failures if their exponential factor β is less than 2.9, and they are more robust against intentional attacks if β is smaller. In the present of cascading failure, we show that power-law networks are very vulnerable when cascading failure occurs since any random failures of high degree nodes can easily overload the low degree nodes.

At last, we study the optimization of power-law networks, from design and protection perspectives. On the one hand, we reveal the best range $[1.8, 2.5]$ for the exponential factor β by optimizing the complex networks in terms of both their vulnerabilities and costs. When $\beta < 1.8$, the network maintenance cost is very expensive, and when $\beta > 2.5$ the network robustness is unpredictable since it depends on the specific attacking strategy. On the other hand, we study *Critical Link Disruptor (CLD)* and *Critical Node Disruptor (CND)* optimization problems to identify critical links and nodes in a network whose removals maximally destroy the network's functions. After showing the NP-hardness of these two problems, we propose HILPR, a novel LP-based

rounding algorithm, for efficiently solving CLD and CND problems in a timely manner. In the case of cascading failures, we further develop the TRPA algorithm, an iterative 2-phase algorithm, for solving *Cascading Critical Node Disruptor (CCND)* problem. The effectiveness of our solutions is validated on various synthetic and real-world networks.

CHAPTER 1
INTRODUCTION

One of the most remarkable discoveries in many real-world networks is the power-law distribution in their degree sequences, ranging from the Internet [34], WWW [4], biological networks [12] to social networks [74]. In particular, the number of nodes with degree i in these complex networks is observed to be proportional to $i^{-\beta}$ for some exponential factor $\beta > 1$.

1.1 Power-Law Graphs

1.1.1 Formal Definition

We consider the following graph, (α, β) *graph* $G_{(\alpha,\beta)}$, with its power-law degree distribution depending on two given values α and β.

Definition 1 ((α, β) Graph $G_{(\alpha,\beta)}$)**.** *Given an undirected graph $G = (V, E)$ having $|V| = n$ nodes and $|E| = m$ edges, it is called a (α, β) power-law graph if its maximum degree is $\Delta = \lfloor e^{\alpha/\beta} \rfloor$ and the number of nodes with degree i is*

$$y_i = \begin{cases} \lfloor \frac{e^\alpha}{i^\beta} \rfloor, & \text{if } i > 1 \text{ or } \sum_{i=1}^{\Delta} \lfloor \frac{e^\alpha}{i^\beta} \rfloor \text{ is even} \\ \lfloor e^\alpha \rfloor + 1, & \text{otherwise} \end{cases} \tag{1–1}$$

Note that the number of nodes $n = e^\alpha \zeta(\beta) + O(n^{\frac{1}{\beta}} - 1)$ and the number of edges $m = \frac{1}{2} e^\alpha \zeta(\beta - 1) + O(n^{\frac{2}{\beta}} - 1)$, where $\zeta(\beta) = \sum_{i=1}^{\infty} \frac{1}{i^\beta}$ is the Riemann Zeta function*. For simplicity, since there is only a very small error $o(1)$ when $\beta > 2$ when counting the number of both nodes and edges, we denote them as $n \doteq e^\alpha \zeta(\beta)$ and edges $m \doteq \frac{1}{2} e^\alpha \zeta(\beta - 1)$.*

1.1.2 Random Power-Law Graph Model

There are two main categories of random graph models to generate graphs with skewed degree sequences, evolutionary and structural. *Evolutionary* models lead to the skewed degree distributions by identifying growth primitives, including multi-objective optimization [6, 33] and statistical preferential attachment [11, 24, 58, 65]. Despite its advantage to explore additional network semantics, the tight dependencies

between iterations in evolutionary models bring the biggest obstacle in the probabilistic analysis [15, 33]. Structural models, on the other hand, start with a given skewed degree distribution (e.g., a power-law distribution based on the degree sequences of a real-world network [18]) and generate a graph with the degree sequence, satisfying certain randomness properties [2, 37, 82]. The greatest advantage of such structural models is their tractability to theoretical analysis, due to its discard of dependencies in evolutionary models by taking skewed degree sequences [2, 21, 66]. Although the term configuration is used, a lot of mathematicians also noted this advantage by exploiting several properties in structural random graph models [14, 68, 69].

Therefore, in this dissertation, we use the well-accepted structural PLRG model in [2] in order to explore the power-law networks from an in-depth theoretical perspective. Given the parameters α and β, the PLRG model is proposed as an structural approach to construct a (α, β) power-law graph according to its degree sequence $\vec{d}$, which consists of a sequence of integers $(1, \ldots, 1, 2, \ldots, 2, \ldots, \Delta)$ where the number of i is equal to y_i defined in the above Definition 1.

Definition 2 (Power-Law Random Graph (PLRG) Model)**.** *Given* $\vec{d} = (d_1, d_2, \ldots, d_n)$ *be a sequence of integers* $(1, \ldots, 1, 2, \ldots, 2, \ldots, \Delta)$ *where the number of* i *is equal to* y_i*, the PLRG model generates a random graph as follows. Consider* $D = \sum_{i=1}^{n} d_i$ *mini-nodes lying in* n *clusters of each size* d_i *where* $1 \leq i \leq n$*, we construct a random perfect matching among the mini-nodes and generate a graph on the* n *original nodes as suggested by this perfect matching in the natural way: two original nodes are connected by an edge if and only if at least one edge in the random perfect matching connects the mini-nodes of their corresponding clusters.*

1.2 Optimization Problems in Power-Law Graphs

A great number of large-scale networks in real life are discovered to follow a power-law distribution in their degree sequences, ranging from the Internet [34], the World-Wide Web (WWW) [4] to social networks [74]. That is, the number of vertices

with degree i is proportional to $i^{-\beta}$ for some constant β in these graphs, which is called power-law graphs. The observations show that the exponential factor β ranges between 1 and 4 for most real-world networks [18]. Intuitively, the following theoretical question is raised: What are the differences in terms of complexity hardness and inapproximability factor of several optimization problems between in general graphs and in power-law graphs?

Many experimental results on random power-law graphs give us a belief that the problems might be much easier to solve on power-law graphs. Eubank *et al.* [32] showed that a simple greedy algorithm leads to a $1 + o(1)$ approximation factor on MINIMUM DOMINATING SET (MDS) and MINIMUM VERTEX COVER (MVC) on power-law graphs (without any formal proof) although MDS and MVC has been proved *NP*-hard to be approximated within $(1 - \epsilon) \log n$ and 1.366 on general graphs respectively [28]. In [73], Gopal also claimed that there exists a polynomial time algorithm that guarantees a $1 + o(1)$ approximation of the MVC problem with probability at least $1 - o(1)$. Unfortunately, there is no such formal proof for this claim either. Furthermore, several papers also have some theoretical guarantees for some problems on power-law graphs. Gkantsidis *et al.* [36] proved the flow through each link is at most $O(n \log^2 n)$ on power-law random graphs where the routing of $O(d_u d_v)$ units of flow between each pair of vertices u and v with degrees d_u and d_v. In [36], the authors take advantage of the property of power-law distribution by using the structural random model [2] and show the theoretical upper bound with high probability $1 - o(1)$ and the corresponding experimental results. Likewise, Janson *et al.* [48] gave an algorithm that approximated MAXIMUM CLIQUE within $1 - o(1)$ on power-law graphs with high probability on the random poisson model $G(n, \alpha)$ (i.e. the number of vertices with degree at least i decreases roughly as n^{-i}). Although these results were based on experiments and various random models, they raise an interest in investigating hardness and inapproximability of optimization problems on power-law graphs.

Recently, Ferrante *et al.* [35] had an initial attempt on power-law graphs to show the *NP*-hardness of MAXIMUM CLIQUE (CLIQUE) and MINIMUM GRAPH COLORING (COLORING) ($\beta > 1$) by constructing a bipartite graph to embed a general graph into a power-law graph and *NP*-hardness of MVC, MDS and MAXIMUM INDEPENDENT SET (MIS) ($\beta > 0$) based on their optimal substructure properties.

1.3 Vulnerability Assessment of Power-Law Networks

Most studies investigating this power-law property have been focused on how such degree heterogeneity nature can impact the robustness of networks [3, 5, 43], or how one can quickly and efficiently generate an ideal power-law network with a given degree sequence [2, 14]. Focusing on the security factor, the works [5, 23, 43, 72] have empirically shown that power-law networks appear robust under random attacks and vulnerable to intentional attacks via experimental observations. Nevertheless, there are several important security aspects of this property that are left untouched. For instance, are power-law networks surely more vulnerable to intentional attacks than random failures? How can we accurately assess the robustness of power-law networks under various kinds of threat, e.g., random failure and adversarial attack? Can we design more stable and robust power-law networks by adjusting the parameter β?

Another limitation of these prior works is their heavy dependence on the experiments and failures to optimize the power-law networks. In other words, we cannot apply them to enhance the robustness of power-law networks, and in the meanwhile reduce their costs. To our best knowledge, this work is the first attempt from a theoretical point of view targeting in the two objectives mentioned above: (1) assessing the impact of random and intentional attacks on power-law networks; (2) optimizing power-law networks based on their toleration on threats and maintenance costs, which are used to guarantee the network functionality and reliability.

1.4 Optimization of Power-Law Networks

Although power-law networks are more robust when β is smaller, a majority of real-world networks usually have their exponential factor β ranging from 2 to 2.5 rather than some small β approaching 1 or even less. The questions are intuitively raised: Is it better if real-world networks are denser such that they can be more robust? What causes them to be sparser than our expectation? Does there exist some potential optimization factors?

On the other hand, in order to optimally maintain the power-law networks, it is of great importance to assess the network vulnerability, that is, to study how much the network performance reduces in various cases of undesired disruptions, such as natural disasters, unexpected elements failures, or especially adversarial attacks. In a typical attacking point of view, an attacker would first exploit the network weaknesses, and then only needs to target on some critical links or nodes whose corruptions bring the whole network down to its knees. For instance, an adversarial attack to any essential Internet providers, e.g., tier-1 ISPs such as Qwest, AT&T or Sprint servers, once successful, may cause tremendous breakdowns to millions of companies' websites and online services. In a natural disaster, an unexpected earthquake may destroy some important power lines, and consequently lead to a large-area blackout. Therefore, it is crucial to explore the network vulnerability, i.e., identify those crucial links and nodes, beforehand.

1.5 Outline of Dissertation

The rest of dissertation, focusing on addressing the above three topics, is organized as follows: Chapter 2 presents the hardness and inapproximability results of classic optimization problem in power-law networks, in which we propose two novel techniques to embed a d-bounded graph into general power-law graphs and simple power-law graphs respectively. In addition, we design a Low-Degree Percolation (LDP) Algorithm Framework for these optimization problems, and further provide a theoretical framework to analyze approximation ratios in power-law graphs. In Chapter 3, we explore the

vulnerability of power-law networks via in-depth probabilistic analysis, under random failures, intentional attacks, and random cascading failures. Chapter 4 investigates the optimization of power-law networks. From a design perspective, we show that, in both communication and social context, the power-law networks with exponential factor between 1.8 and 2.5 results in the optimal design. Furthermore, in order to better protect the power-law networks, we study CLD, CND and CCND problems to detect critical elements. After showing the NP-hardness of these problems, we develop HILPR and TRGA algorithms to solve them in a timely manner. The whole dissertation is concluded in Chapter 5.

CHAPTER 2
HARDNESS AND APPROXIMATION ALGORITHMS

In this chapter, we develop two new techniques on optimal substructure problems, *Cycle-Based Embedding Technique* and *Graphic Embedding Technique*, to embed a d-bounded graph into a general power-law graph and a simple power-law graph respectively. Then we use these two techniques to further prove the *APX*-hardness and the inapproximability of MIS, MDS, and MVC on general power-law graphs and simple power-law graphs. These inapproximability results on power-law graphs are shown in Table 2-1. Furthermore, the inapproximability results in CLIQUE and COLORING are shown by taking advantage of the reduction in [35]. We also analyze the relationship between β and constant greedy approximation algorithms for MIS and MDS.

In addition, due to a lot of recent studies in online social networks on the influence propagation problem [54, 55], we formulate this problem as ρ-*Minimum Dominating Set* (ρ-MDS) and show it hard to be approximated within $2-(2+o_d(1))\log\log d/\log d$ factor on d-bounded graphs under unique games conjecture, which further leads to the following inapproximability result on power-law graphs (shown in Table 2-1).

Table 2-1. Inapproximability Factors on Power-Law Graphs with Exponential Factor $\beta > 1$

Problem	General Power-Law Graph	Simple Power-Law Graph
MIS	$1+\frac{1}{140(2\zeta(\beta)3^{\beta}-1)}-\varepsilon$	$1+\frac{1}{1120\zeta(\beta)3^{\beta}}-\varepsilon$
MDS	$1+\frac{1}{390(2\zeta(\beta)3^{\beta}-1)}$	$1+\frac{1}{3120\zeta(\beta)3^{\beta}}$
MVC, ρ-MDS	$1+\frac{2\left(1-(2+o_c(1))\frac{\log\log c}{\log c}\right)}{\left(\zeta(\beta)c^{\beta}+c^{\frac{1}{\beta}}\right)(c+1)}$	$1+\frac{2-(2+o_c(1))\frac{\log\log c}{\log c}}{2\zeta(\beta)c^{\beta}(c+1)}$
CLIQUE	-	$O\left(n^{1/(\beta+1)-\epsilon}\right)$
COLORING	-	$O\left(n^{1/(\beta+1)-\epsilon}\right)$

[a] Conditions: MIS and MDS: P≠NP; MVC, ρ-MDS: unique games conjecture; CLIQUE, COLORING: NP≠ZPP.
[b] c is a constant which is the smallest d satisfying the condition in [10].

2.1 Preliminaries

In this section, we first recall the definition of several classical optimization problems and formulate the new optimization problem ρ-Minimum Dominating Set. Then the

power-law model and some corresponding concepts are proposed. At last, we introduce some special graphs which will be used in the analysis throughout the whole paper.

2.1.1 Problem Definitions

Definition 3 (MAXIMUM INDEPENDENT SET)**.** *Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ with the maximum size such that no two vertices in S are adjacent.*

Definition 4 (MINIMUM VERTEX COVER)**.** *Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ with the minimum size such that for each edge E at least one endpoint belongs to S.*

Definition 5 (MINIMUM DOMINATING SET)**.** *Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ with the minimum size such that for each vertex $v_i \in V \setminus S$, at least one neighbor of v_i belongs to S.*

Definition 6 (MAXIMUM CLIQUE)**.** *Given an undirected graph $G = (V, E)$, find a clique with maximum size where a subgraph of G is called a clique if all its vertices are pairwise adjacent.*

Definition 7 (MINIMUM GRAPH COLORING)**.** *Given an undirected graph $G = (V, E)$, label the vertices in V with minimum number of colors such that no two adjacent vertices share the same color.*

The ρ-Minimum Dominating Set is defined as general version of MDS problem. In the context of influence propagation, the ρ-MDS problem aims to find a subset of nodes with minimum size such that all nodes in the whole network can be influenced within t rounds. In particular, a node is influenced when ρ fraction of its neighbors are influenced. For simplicity, we define ρ-MDS problem in the case that $t = 1$.

Definition 8 (ρ-MINIMUM DOMINATING SET)**.** *Given an undirected graph $G = (V, E)$, find a subset $S \subseteq V$ with the minimum size such that for each vertex $v_i \in V \setminus S$, $|S \cap N(v_i)| \geq \rho |N(v_i)|$.*

2.1.2 Some Notations

A great number of models [2, 2, 11, 13, 71] on power-law graphs are emerging in the past recent years. In this chapter, we do the analysis based on the general (α, β) model, that is, the graphs only constrained by the power-law distribution in degree sequences. We first define the following two types of degree sequences.

Definition 9 (y-Degree Sequence)**.** *Given a graph $G = (V, E)$, the y-degree sequence of G is a sequence $Y = \langle y_1, y_2, \dots, y_\Delta \rangle$ where Δ is the maximum degree of G and $y_i = |\{u | u \in V \wedge deg(u) = i\}|$.*

Definition 10 (d-Degree Sequence)**.** *Given a graph $G = (V, E)$, the d-degree sequence of G is a sequence $D = \langle d_1, d_2, \dots, d_n \rangle$ of vertex in non-increasing order of their degrees.*

Note that y-degree sequence and d-degree sequence are interchangeable. Given a y-degree sequence $Y = \langle y_1, y_2, \dots, y_\Delta \rangle$, the corresponding d-degree sequence is $D = \langle \Delta, \Delta, \dots, \Delta - 1, \Delta - 1, \dots, \Delta - 1, \dots, 1, \dots, 1 \rangle$ where the number i appears y_i times. Because of their equivalence, we may use only y-degree sequence or d-degree sequence or both without changing the meaning or validity of results.

Definition 11 (Continuous Sequence)**.** *An integer sequence $\langle d_1, d_2, \dots, d_n \rangle$, where $d_1 \geq d_2 \geq \cdots \geq d_n$, is continuous if $\forall 1 \leq i \leq n - 1$, $|d_i - d_{i+1}| \leq 1$.*

Definition 12 (Graphic Sequence)**.** *A sequence D is said to be graphic if there exists a graph such that D is its d-degree sequence.*

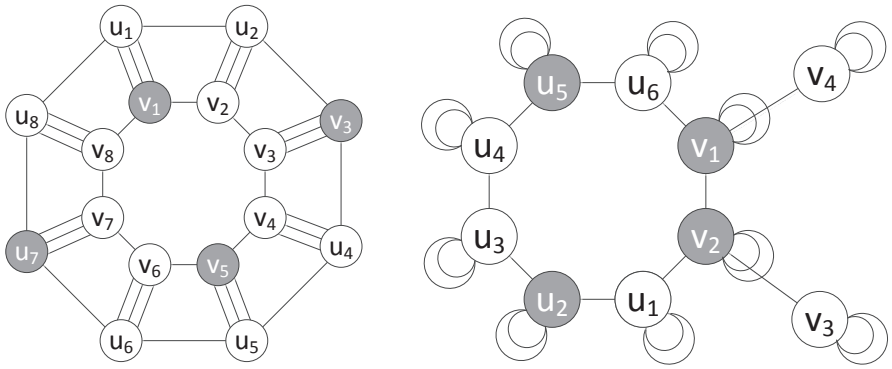
Definition 13 (Degree Set)**.** *Given a graph G, let $D_i(G)$ be the set of vertices of degree i on G.*

Furthermore, we define the d-bounded graph as

Definition 14 (d-Bounded Graph)**.** *Given a graph $G = (V, E)$, G is a d-bounded graph if the degree of any vertex is upper bounded by an integer constant d.*

2.1.3 Special Graphs

Definition 15 ($\vec{d}$-Regular Cycle $RC_n^{\vec{d}}$)**.** *Given a vector $\vec{d} = (d_1, \dots, d_n)$, a $\vec{d}$-regular cycle RC_n^d is composed of two cycles. Each cycle has n vertices and two i^{th} vertices in*



A RC_8^3 B $\vec{2} - C_8^2$

Figure 2-1. Special Graph Examples: The left one is a $(3,3,3,3,3,3,3,3)$-regular cycle and the right one is a $(3,3,3,3)$-branch-$(2,2,2,2,2,2)$-cycle. The grey vertices consist of the optimal solution of MDS on these two special graphs.

each cycle are adjacent with each other by $d_i - 2$ multi-edges. That is, $\vec{d}$-regular cycle RC_n^d has $2n$ vertices and the two i^{th} vertex has the same degree d_i. An example RC_8^d is shown in Figure 2-1A.

Definition 16 ($\vec{\kappa}$-Branch-$\vec{d}$-Cycle $\vec{\kappa}$-$BC_n^{\vec{d}}$)**.** *Given two vectors $\vec{d} = (d_1, \dots, d_n)$ and $\vec{\kappa} = (\kappa_1, \dots, \kappa_m)$, the $\vec{\kappa}$-branch-$\vec{d}$-cycle is composed of a cycle with a number of vertices n such that each vertex has degree d_i as well as $|\vec{\kappa}|/2$ appendant branches, where $|\kappa|$ is a even number. Note that any $\vec{\kappa}$-branch-$\vec{d}$-cycle has $|\vec{\kappa}|$ even number of vertices with odd degrees. An example is shown in Figure 2-1B.*

2.1.4 Existing Inapproximability Results

Here we list some inapproximability results in the literature to use later in our proofs.

(1) In d-bounded graphs, MVC is hard to be approximated into $2-(2+o_d(1))\log\log d/\log d$ for every sufficiently large integer d under unique games conjecture [10, 20].

(2) In 3-bounded graphs, MIS and MDS is *NP*-hard to be approximated into $\frac{140}{139} - \varepsilon$ for any $\varepsilon > 0$ and $\frac{391}{390}$ respectively [8].

(3) Maximum clique and minimum coloring problem is hard to be approximated into $n^{1-\epsilon}$ on general graphs unless NP=ZPP [41].

2.2 Inapproximability Optimal Substructure Framework in Power-Law Graphs

In this section, we introduce a framework to derive the approximation hardness of optimal substructure problems on power-law graphs. A graph optimization problem is said to satisfy optimal substructure if its optimal solution is the union of the optimal solutions on each connected component. Therefore, when a graph G is embedded into a power-law graph G', the optimal solution in G' consists of a subset of the optimal solution in G. According to this important property, we present the *Inapproximability Optimal Substructure Framework* to prove the inapproximability factor if there exists a *Embedded-Approximation-Preserving Reduction* that relates the approximation hardness in general graphs and power-law graphs by guaranteeing the relationship between the solutions in the original graph and the constructed graph.

Definition 17 (Embedded-Approximation-Preserving Reduction)**.** *Given an optimal substructure problem O, a reduction from an instance on graph $G = (V, E)$ to another instance on a power-law graph $G' = (V', E')$ is called embedded-approximation-preserving if it satisfies the following properties:*

(1) G is a subset of maximal connected components of G';

(2) The optimal solution of O on G', $OPT(G')$, is upper bounded by $\mathfrak{C}OPT(G)$ where $\mathfrak{C}$ is a constant correspondent to the growth of the optimal solution.

Theorem 2.1 (Inapproximability Optimal Substructure Framework)**.** *Given an optimal substructure problem O, if there exists an embedded-approximation-preserving reduction from a graph G to another graph G', we can extract the inapproximability factor δ of O on G' using* ϵ-inapproximability *of O on G, where δ is lower bounded by $\frac{\epsilon\mathfrak{C}}{(\mathfrak{C}-1)\epsilon+1}$ and $\frac{\epsilon+\mathfrak{C}-1}{\mathfrak{C}}$ when O is a maximum and minimum optimization problem respectively.*

Proof. Suppose that there exists an algorithm providing a solution of O on G' with size at most δ times the optimal solution. Denote A and B to be the sizes of the produced solution on G and $G' \setminus G$ and A^* and B^* to be their corresponding optimal values. Hence, we have $B^* \leq (\mathfrak{C}-1)A^*$. With the completeness that $OPT(G) = A^* \Rightarrow OPT(G') = B^*$,

the soundness leads to the lower bound of δ which is dependent on the type of O, maximization or minimization problem, as follows.

Case 1: When O is a maximization problem, we start from the definition of soundness as

$$A^* + B^* \leq \delta(A + B) \tag{2–1}$$

$$\Leftrightarrow A^* \leq \delta A + (\delta - 1)B^* \tag{2–2}$$

$$\Leftrightarrow A^* \leq \delta A + (\delta - 1)(\mathfrak{C} - 1)A^* \tag{2–3}$$

where (2–2) holds since $B \leq B^*$ and (2–3) holds since $B^* \leq (\mathfrak{C} - 1)A^*$.

On the other hand, it is hard to approximate O within ϵ on G, thus $A^* > \epsilon A$. Replace it to the above inequality, we have:

$$A^* < A^*\delta/\epsilon + (\delta - 1)(\mathfrak{C} - 1)A^* \Leftrightarrow \delta > \frac{\epsilon\mathfrak{C}}{(\mathfrak{C} - 1)\epsilon + 1}$$

Case 2: When O is a minimization problem, since $B^* \leq B$, similarly

$$A + B \leq \delta(A^* + B^*)$$

$$\Leftrightarrow A \leq \delta A^* + (\delta - 1)B^*$$

$$\Leftrightarrow A \leq \delta A^* + (\delta - 1)(\mathfrak{C} - 1)A^*$$

Then from $A > \epsilon A^*$,

$$\epsilon < \delta + (\delta - 1)(\mathfrak{C} - 1) \Leftrightarrow \delta > \frac{\epsilon + \mathfrak{C} - 1}{\mathfrak{C}}$$

□

2.3 Hardness and Inapproximability of Optimal Substructure Problems

2.3.1 General Cycle-Based Embedding Technique

In this section, we propose a *General Cycle-Based Embedding Technique* on (α, β) power-law graphs with $\beta > 1$. The basic idea is to embed an arbitrary d-bounded graph

into power-law graphs using a $\vec{d_1}$-regular cycle, a $\vec{\kappa}$-branch-$\vec{d_2}$-cycle and a number of cliques K_2, where $\vec{d_1}$, $\vec{d_2}$ and $\vec{\kappa}$ are defined by α and β. Before discussing the main embedding technique, we first show that most optimal substructure problems can be polynomially solved in both $\vec{d}$-regular cycles and $\vec{\kappa}$-branch-$\vec{d}$-cycle. In this context, the cycle-based embedding technique helps to prove the complexity of these optimal substructure problems on power-law graphs according to their corresponding complexity results on general bounded graphs.

Lemma 1. *MDS, MVC and MIS are polynomially solvable on $\vec{d}$-regular cycles.*

Proof. Here we just prove MDS problem is polynomially solvable on $\vec{d}$-regular cycles. The algorithm is simple. From an arbitrarily vertex, we select the vertex on the other cycle in two hops. The algorithm will terminate until all vertices are dominated. Now we will show that this gives the optimal solution. Let$^t ake$RC$_8^3$ as an example. As shown in Figure 2-1A, the size of MDS is 4. Notice that each vertex can dominate exact 3 vertices, that is, 4 vertices can dominate exactly 12 vertices. However, in RC_8^3, there are altogether 16 vertices, which have to be dominated by at least 4 vertices apart from the vertices in MDS. That is, the algorithm returns an optimal solution. The proof of MVC and MIS is similar. □

Lemma 2. *MDS, MVC and MIS is polynomially solvable on $\vec{\kappa}$-branch-$\vec{d}$-cycles.*

Proof. Again we show the proof of MDS. First we select the vertices connecting both the branches and the cycle. Then by removing the branches, we will have a line graph regardless of self-loops, on which MDS is polynomially solvable. It is easy to see that the size of MDS will increase if any one vertex connecting both the branch and the cycle in MDS is replaced by some other vertices. The proof of MIS is similar. Note that the optimal solution for MVC consists of all vertices since all edges need to be covered. □

Theorem 2.2 (Cycle-Based Embedding Technique)**.** *Any d-bounded graph G_d can be embedded into a power-law graph $G_{(\alpha,\beta)}$ with $\beta > 1$ such that G_d is a maximal*

component and most optimal substructure problems can be polynomially solvable on $G_{(\alpha,\beta)} \setminus G_d$.

Proof. With the given β, we choose α to be $\max\{\ln \max_{1\le i\le d}\{n_i \cdot i^\beta\}, \beta \ln d\}$. Based on $\tau(i) = \lfloor e^\alpha / i^\beta \rfloor - n_i$ where $n_i = 0$ when $i > d$, we construct the power-law graph $G_{(\alpha,\beta)}$ as the following Algorithm 1. The last step holds since the number of vertices of odd degrees has to be even. From Step 1, we know $e^\alpha = \max\{\max_{1\le i\le d}\{n_i \cdot i^\beta\}, d^\beta\} \le d^\beta n$, that is, the number of vertices N in graph $G_{(\alpha,\beta)}$ satisfies $N \le \zeta(\beta) d^\beta n$, which means that N/n is a constant. According to Lemma 1 and Lemma 2, since $G_{(\alpha,\beta)} \setminus G_d$ is composed of a $\vec{d_1}$-regular cycle and a $\vec{d_2^1}$-branch-$\vec{d_2}$-cycle, it can be polynomially solvable. Note that the number of vertices in L is at most Δ since there is at most one leftover vertex of each degree.

Algorithm 1: Cycle Embedding Algorithm

1. $\alpha \leftarrow \max\{\ln \max_{1\le i\le d}\{n_i \cdot i^\beta\}, \beta \ln d\}$;
2. For $\tau(1)$ vertices of degree 1, add $\lfloor \tau(1)/2 \rfloor$ number of cliques K_2;
3. For $\tau(2)$ vertices of degree 2, add a cycle with the size $\tau(2)$;
4. For all vertices of degree larger than 2 and smaller than Δ, construct a $\vec{d_1}$-regular cycle where $\vec{d_1}$ is a vector composed of $\lfloor \tau(i)/2 \rfloor$ number of elements i for all i satisfying $\tau(i) > 0$;
5. For all leftover isolated vertices L such that $\tau(i) - 2\lfloor \tau(i)/2 \rfloor = 1$, construct a $\vec{d_2^1}$-branch-$\vec{d_2^2}$-cycle, where $\vec{d_2^1}$ and $\vec{d_2^2}$ are the vectors containing odd and even elements correspondent to the vertices of odd and even degrees in L respectively.

□

2.3.2 *APX*-Hardness

In this section, we prove that MIS, MDS, MVC remain *APX*-hard even on power-law graphs.

Theorem 2.3. *MDS is* APX*-hard on power-law graphs.*

Proof. According to Theorem 2.2, we use the cycle-based embedding technique to show $\mathcal{L}$-reduction from MDS on any d-bounded graph G_d to MDS on a power-law graph $G_{(\alpha,\beta)}$ since MDS is proven *APX*-hard on d-bounded graphs [51].

Letting ϕ be a feasible solution on G_d, we can construct MDS in G' such that MDS on a K_2 is 1, $n/4$ on a $\vec{d}$-regular cycle and $n/3$ on a cycle and a $\vec{\kappa}$-branch-$\vec{d}$-cycle. Therefore, for a solution ϕ on G_d, we have a solution φ on $G_{(\alpha,\beta)}$ to be $\varphi = \phi + n_1/2 + n_2/3 + n_3/4$, where n_1, n_2 and n_3 corresponds to $\tau(1)$, $\tau(2) \cup L$ and all leftover vertices. Hence, we have $OPT(\varphi) = OPT(\phi) + n_1/2 + n_2/3 + n_3/4$.

On one hand, for a d-bounded graph with vertices n, the optimal MDS is lower bounded by $n/(d+1)$. Thus, we know

$$\begin{aligned} & OPT(\varphi) = OPT(\phi) + n_1/2 + n_2/3 + n_3/4 \\ \leq\ & OPT(\phi) + (N-n)/2 \leq OPT(\phi) + (\zeta(\beta)d^\beta - 1)n/2 \\ \leq\ & OPT(\phi) + (\zeta(\beta)d^\beta - 1)(d+1)OPT(\phi)/2 = \left[1 + (\zeta(\beta)d^\beta - 1)(d+1)/2\right] OPT(\phi) \end{aligned}$$

where N is the number of vertices in $G_{(\alpha,\beta)}$.

On the other hand, with $|OPT(\phi) - \phi| = |OPT(\varphi) - \varphi|$, we proved the $\mathcal{L}$-reduction with $c_1 = 1 + (\zeta(\beta)d^\beta - 1)(d+1)/2$ and $c_2 = 1$. □

Theorem 2.4. *MVC is* APX*-hard on power-law graphs.*

Proof. In this proof, we show $\mathcal{L}$-reduction from MVC on d-bounded graph G_d to MVC on power-law graph $G_{(\alpha,\beta)}$ using cycle-based embedding technique.

Let ϕ be a feasible solution on G_d. We construct the solution $\varphi \leq \phi + (N - n)$ since the optimal solution of MVC is $n/2$ on K_2, cycle, $\vec{d}$-regular cycle and n on $\vec{\kappa}$-branch-$\vec{d}$-cycle. Therefore, since the optimal MVC on a d-bounded graph is lower bounded by $n/(d+1)$, we have

$$OPT(\varphi) \leq \left[1 + (\zeta(\beta)d^\beta - 1)(d+1)\right] OPT(\phi)$$

On the other hand, with $|OPT(\phi) - \phi| = |OPT(\varphi) - \varphi|$, we proved the $\mathcal{L}$-reduction with $c_1 = 1 + (\zeta(\beta)d^\beta - 1)(d+1)$ and $c_2 = 1$. □

Corollary 1. *MIS is* APX*-hard on power-law graphs.*

2.3.3 Inapproximability Factors

In this section, we show the inapproximability factors on MIS, MVC and MDS on power-law graphs respectively using the results in section 2.1.4.

Theorem 2.5. *For any $\varepsilon > 0$, there is no $1 + \frac{1}{140(2\zeta(\beta)3^\beta - 1)} - \varepsilon$ approximation algorithm for Maximum Independent Set on power-law graphs.*

Proof. In this proof, we construct the power-law graph $G_{(\alpha,\beta)}$ based on cycle-based embedding technique in Theorem 2.2 from d-bounded graph G_d. Let ϕ and φ be feasible solutions of MIS on G_d and $G_{(\alpha,\beta)}$. Then $OPT(\varphi)$ composed of $OPT(\phi)$, clique K_2, cycle, $\vec{d}$-regular cycle and $\vec{\kappa}$-branch-$\vec{d}$-cycles are all exactly half number of vertices. Hence, we have $OPT(\varphi) = OPT(\phi) + (N - n)/2$ where n and N is the number of vertices in G_d and $G_{(\alpha,\beta)}$ respectively. Since $OPT(\phi) \geq n/(d+1)$ on d-bounded graphs for MIS and $N \leq \zeta(\beta)d^\beta n$, we further have $\mathfrak{C} = 1 + \frac{(\zeta(\beta)d^\beta - 1)(d+1)}{2}$ from

$$\begin{aligned} OPT(\varphi) &= OPT(\phi) + \frac{N-n}{2} \leq OPT(\phi) + \frac{(\zeta(\beta)d^\beta - 1)}{2} n \\ &\leq OPT(\phi) + \frac{(\zeta(\beta)d^\beta - 1)(d+1)}{2} OPT(\phi) \\ &= \left(1 + \frac{(\zeta(\beta)d^\beta - 1)(d+1)}{2}\right) OPT(\phi) \end{aligned}$$

According to $\epsilon = \frac{140}{139} - \varepsilon'$ for any $\epsilon' > 0$ on 3-bounded graphs, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

$$\delta > \frac{\epsilon\mathfrak{C}}{(\mathfrak{C}-1)\epsilon + 1} > 1 + \frac{1}{140\mathfrak{C}} - \varepsilon = 1 + \frac{1}{140(2\zeta(\beta)3^\beta - 1)} - \varepsilon$$

where the last step follows from $d = 3$.

□

Theorem 2.6. *There is no* $1 + \frac{1}{390(2\zeta(\beta)3^{\beta}-1)}$ *approximation algorithm for Minimum Dominating Set on power-law graphs.*

Proof. In this proof, we construct the power-law graph $G_{(\alpha,\beta)}$ based on cycle-based embedding technique in Theorem 2.2 from d-bounded graph G_d. Let ϕ and φ be feasible solutions of MDS on G_d and $G_{(\alpha,\beta)}$. The optimal MDS on $OPT(\phi)$, clique K_2, cycle, $\vec{d}$-regular cycle and $\vec{\kappa}$-branch-$\vec{d}$-cycles are $n/2$, $n/4$ and $n/3$ respectively. Let ϕ and φ be feasible solutions of MDS on G_d and $G_{(\alpha,\beta)}$. Then we have $\mathfrak{C} = 1 + \frac{(\zeta(\beta)d^{\beta}-1)(d+1)}{2}$ similar as the proof in Theorem 2.5.

According to $\epsilon = \frac{391}{390}$ in 3-bounded graphs, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

$$\delta > 1 + \frac{\epsilon - 1}{\mathfrak{C}} = 1 + \frac{1}{390(2\zeta(\beta)3^{\beta} - 1)}$$

where the last step follows from $d = 3$.

□

Theorem 2.7. *MVC is hard to be approximated within* $1 + \frac{2\left(1-(2+o_c(1))\frac{\log\log c}{\log c}\right)}{\left(\zeta(\beta)c^{\beta}+c^{\frac{1}{\beta}}\right)(c+1)}$ *on power-law graphs under unique games conjecture.*

Proof. By constructing the power-law graph $G_{(\alpha,\beta)}$ based on cycle-based embedding technique in Theorem 2.2 from d-bounded graph G_d, The optimal MVC on clique K_2, cycle, $\vec{d}$-regular cycle are half number of vertices while the optimal MVC on $\vec{\kappa}$-branch-$\vec{d}$-cycles are all vertices. Thus, we have $\mathfrak{C} = 1 + \frac{\left(\zeta(\beta)d^{\beta}-1+d^{\frac{1}{\beta}}\right)(d+1)}{2}$ since

$$
\begin{aligned}
OPT(\varphi) &\leq OPT(\phi) + \frac{N - n - \Delta}{2} + \Delta \leq OPT(\phi) + \frac{(\zeta(\beta)d^{\beta} - 1)n + n^{\frac{1}{\beta}}d}{2} && (2\text{–}4)\\
&= OPT(\phi) + \frac{\left(\zeta(\beta)d^{\beta} - 1 + \frac{d}{n^{1-\frac{1}{\beta}}}\right)n}{2} && (2\text{–}5)\\
&\leq OPT(\phi) + \frac{\left(\zeta(\beta)d^{\beta} - 1 + \frac{d}{(d+1)^{1-\frac{1}{\beta}}}\right)(d+1)}{2} OPT(\phi) && (2\text{–}6)\\
&\leq \left(1 + \frac{\left(\zeta(\beta)d^{\beta} - 1 + d^{\frac{1}{\beta}}\right)(d+1)}{2}\right) OPT(\phi) && (2\text{–}7)
\end{aligned}
$$

where ϕ and φ be feasible solutions of MVC on G_d and $G_{(\alpha,\beta)}$, Δ is the maximum degree in $G_{(\alpha,\beta)}$. The inequality (2–4) holds since there are at most Δ vertices in $\vec{\kappa}$-branch-$\vec{d}$-cycle, i.e. $\Delta = e^{\alpha/\beta} \leq n^{1/\beta}d$; (2–6) holds since there are at least $d+1$ vertices in a d-bounded graph and the optimal MVC in a d-bounded graph is at least $n/(d+1)$.

According to $\epsilon = 2 - (2 + o_d(1))\log\log d/\log d$, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

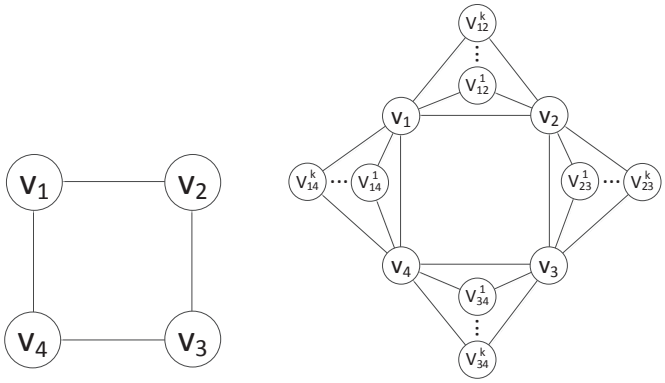
$$
\delta > 1 + \frac{\epsilon - 1}{\mathfrak{C}} \geq 1 + \frac{2\left(1 - (2 + o_c(1))\frac{\log\log c}{\log c}\right)}{\left(\zeta(\beta)c^{\beta} + c^{\frac{1}{\beta}}\right)(c+1)}
$$

where c is the smallest d satisfying the condition in [10]. The last inequality holds since function $f(x) = (1 - (2 + o_x(1))\log\log x/\log x)/g(x)(x+1)$ is monotonously decreasing when $f(x) > 0$ for all $x > 0$ when $g(x)$ is monotonously increasing. □

Theorem 2.8. *ρ-PDS is hard to be approximated into $2 - (2 + o_d(1))\frac{\log\log d}{\log d}$ on d-bounded graphs under unique games conjecture.*

Proof. In this proof, we show the gap-preserving from MVC on (d/ρ)-bounded graph $G = (V, E)$ to ρ-PDS on d-bounded graph $G' = (V', E')$. *w.l.o.g.*, we assume that d and d/ρ are integers. We construct a graph $G' = (V', E')$ by adding new vertices and

edges to G as follows. For each edge $(v_i, v_j) \in E$, create k new vertices $v_{ij}^1, \ldots, v_{ij}^k$ where $1 \leq k \leq \lfloor 1/\rho \rfloor$ and $\rho \leq 1/2$. Then we add $2k$ new edges (v_{ij}^l, v_i) and (v_{ij}^l, v_j) for all $l \in [1, k]$ as shown in Figure 2-2. Clearly, $G' = (V', E')$ is a d-bounded graph.



A Instance $G = (V, E)$ B Reduced Instance $G' = (V', E')$

Figure 2-2. The Reduction from MVC to ρ-MDS

Let ϕ and φ be feasible solutions to MVC on G and G' respectively. We claim that $OPT(\phi) = OPT(\varphi)$.

On one hand, if $S = \{v_1, v_2, \ldots, v_j\} \in V$ is the minimum vertex cover on G. Then $\{v_1, v_2, \ldots, v_j\}$ is a ρ-PDS on G' because each vertex in V has ρ of all neighbors in MVC and every new vertex in $V' \setminus V$ has at least one of two neighbors in MVC. Thus $OPT(\phi) \geq OPT(\varphi)$. One the other hand, we can prove that $OPT(\varphi)$ does not contain new vertices, that is, $V' \setminus V$. Consider a vertex $v_i \in V$, if $v_i \in OPT(\varphi)$, the new vertices v_{ij}^l for all $v_j \in N(v_i)$ and all $l \in [1, k]$ are not needed to be selected. If $v_i \notin OPT(\varphi)$, it has to be dominated by ρ proportion of its all neighbors. That is, for each edge (v_i, v_j) incident to v_i, either v_j or all v_{ij}^l have to be selected since every v_{ij}^l has to be either selected or dominated. If all v_{ij}^l are selected in $OPT(\varphi)$ for some edge (v_i, v_j), v_j is still not dominated by enough vertices if there are some more edges incident to v_j and the number of vertices v_{ij}^l k is great than 1, that is, $\lfloor 1/\rho \rfloor \geq 1$. In this case, v_j will be selected to dominate all v_{ij}^l. Thus, $OPT(\varphi)$ does not contain new vertices. Since the vertices in V selected is a solution to ρ-MDS, that is, for each vertex v_i in graph G, v_i will be selected

or at least the number of neighbors of v_i will be selected. Therefore, the vertices in $OPT(\varphi)$ consist of a vertex cover in G. Thus $OPT(\phi) \leq OPT(\varphi)$. Then we show the completeness and soundness as follows.

- If $OPT(\phi) = m \Rightarrow OPT(\varphi) = m$
- If $OPT(\phi) > \left(2 - (2 + o_d(1))\frac{\log\log(d/2)}{\log(d/2)}\right) m \Rightarrow OPT(\varphi) > \left(2 - (2 + o_d(1))\frac{\log\log d}{\log d}\right) m$

$$OPT(\varphi) > \left(2 - (2 + o_d(1))\frac{\log\log(d/\rho)}{\log(d/\rho)}\right) m > \left(2 - (2 + o_d(1))\frac{\log\log d}{\log d}\right) m$$

since the function $f(x) = 2 - \log\log x/\log x$ is monotonously increasing for any $x > 0$.

□

Theorem 2.9. *ρ-PDS is hard to be approximated into $1 + \frac{2\left(1-(2+o_c(1))\frac{\log\log c}{\log c}\right)}{2+(\zeta(\beta)c^\beta-1)(c+1)}$ on power-law graphs under unique games conjecture.*

Proof. By constructing the power-law graph $G_{(\alpha,\beta)}$ based on cycle-based embedding technique in Theorem 2.2 from d-bounded graph G_d, According to the optimal MVC on $OPT(\phi)$, clique K_2, cycle, $\vec{d}$-regular cycle and $\vec{\kappa}$-branch-$\vec{d}$-cycles, we have $\mathfrak{C} = 1 + \frac{(\zeta(\beta)d^\beta-1)(d+1)}{2}$ from

$$\begin{aligned} OPT(\varphi) &= OPT(\phi) + n_1/2 + f(\rho)n_2 + g(\rho)n_3 \\ &\leq OPT(\phi) + \frac{N-n}{2} \leq \left(1 + \frac{(\zeta(\beta)d^\beta - 1)(d+1)}{2}\right) OPT(\phi) \end{aligned}$$

where $f(\rho) = \begin{cases} \frac{1}{4}, & \rho \leq \frac{1}{3} \\ \frac{1}{3}, & \frac{1}{3} < \rho \leq \frac{1}{2} \end{cases}$, $g(\rho) = \frac{1}{3}$ for all $\rho \leq \frac{1}{2}$ and ϕ, φ be feasible solutions of MVC on G_d and $G_{(\alpha,\beta)}$. n_1, n_2 and n_3 are correspondent to the number of vertices in cliques K_2, cycle, $\vec{d}$-regular cycle and $\vec{\kappa}$-branch-$\vec{d}$-cycle.

According to $\epsilon = 2 - (2 + o_d(1)) \log\log d / \log d$, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

$$\delta > 1 + \frac{\epsilon - 1}{\mathfrak{C}} \geq 1 + \frac{2\left(1 - (2 + o_c(1))\frac{\log\log c}{\log c}\right)}{2 + (\zeta(\beta)c^{\beta} - 1)(c + 1)}$$

where c is the smallest d satisfying the condition in [10]. The last inequality holds since function $f(x) = (1 - (2 + o_x(1)) \log\log x / \log x)/g(x)(x + 1)$ is monotonously decreasing when $f(x) > 0$ for all $x > 0$ when $g(x)$ is monotonously increasing.

□

2.4 More Inapproximability Results on Simple Power-Law Graphs

2.4.1 General Graphic Embedding Technique

In this section, we introduce a general graphic embedding technique to embed a d bounded graph into a simple power-law graph. Before presenting the embedding technique, we first show that a graph can be constructed in polynomial time from a class of integer sequences.

Lemma 3. *Given a sequence of integers $D = \langle d_1, d_2, \ldots, d_n \rangle$ which is non-increasing, continuous and the number of elements is at least as twice as the largest element in D, i.e. $n \geq 2d_1$, it is possible to construct a simple graph G whose d-degree sequence is D in polynomial time $O(n^2 \log n)$.*

Proof. Starting with a set of individual vertices S of degree 0 and $|S| = n$, we iteratively connect vertices together to increase their degrees up to given degree sequence. In each step, the leftover vertex of highest degree is connected to other vertices one by one in the decreasing order of their degrees. Then the sequence D will be resorted and all zero elements will be removed. The algorithm stops until D is empty. The whole algorithm is shown as follows (Algorithm 2).

After each while loop, the new degree sequence, called D', is still continuous and its number of elements is at least as twice as its maximum element. To show this, we

Algorithm 2: Graphic Sequence Construction Algorithm

Input : d-degree sequence $D = \langle d_1, d_2, \dots, d_n \rangle$ where $d_1 \geq d_2 \geq \dots \geq d_n$
Output: Graph H

```
while D ≠ ∅ do
    Connect vertex of d_1 to vertices of d_2, d_3, ... , d_{d_1+1};
    d_1 ← 0;
    for i = 2 to d_1 + 1 do
        d_i ← d_i − 1;
    end
    Sort D in non-increasing order;
    Remove all zero elements in D;
end
```

consider three cases: (1) If the maximum degree in D' remains the same, there are at least $d_1 + 2$ vertices in D. Since D is continuous, the number of elements in D is at least $d_1 + 2 + d_1 - 1$, that is, $2d_1 + 1$. Therefore, the number of elements in D' is $2d_1$, i.e. $n \geq 2d_1$ still holds. (2) If the maximum degree in D' is decreased by 1, there are at least 2 elements of degree d_1 in D. Thus, at most one element in D will become 0. Then we have $n \geq 2d_1 - 2 = 2(d_1 - 1)$. (3) If the maximum degree in D' is decreased by 2, there are at most two element in D becoming 0. Thus, $n \geq 2d_1 - 3 > 2(d_1 - 2)$.

The time complexity of the algorithm is $O(n^2 \log n)$ since there are at most n iterations and each iteration takes at most $O(n \log n)$ to sort the new sequence D. □

Theorem 2.10 (Graphic Embedding Technique)**.** *Any d-bounded graph G_d can be embedded into a simple power-law graph $G_{(\alpha,\beta)}$ with $\beta > 1$ in polynomial time such that G_d is a maximal component and the number of vertices in $G_{(\alpha,\beta)}$ can be polynomially bounded by the number of vertices in G_d.*

Proof. Given a d-bounded degree graph $G_d = (V, E)$ and $\beta > 1$, we construct a power-law graph $G_{(\alpha,\beta)}$ of exponential factor β which includes G_d as a set of maximal components. The construction is shown as Algorithm 3.

Algorithm 3: Graphic Embedding Algorithm

1 $\alpha \leftarrow \max\{\frac{\beta}{\beta-1}(\ln 4 + \beta \ln d), \ln 2 + \ln n + \beta \ln d\}$ and corresponding $G_{(\alpha,\beta)}$;
2 D be the d-degree sequence of $G_{(\alpha,\beta)} \setminus G_d$;
3 Construct $G_{(\alpha,\beta)} \setminus G_d$ using Algorithm 2.

According to the lemma 3, the above construction is valid and finishes in polynomial time. Then we show that N is upper bounded by $\zeta(\beta)2d^\beta n$, where n and N are the number of vertices in G_d and $G_{\alpha,\beta}$ respectively. From the construction, we know either

$$\alpha \geq \frac{\beta}{\beta-1}(\ln 4 + \beta \ln d) \Rightarrow \alpha \geq \ln 4 + \beta \ln d + \alpha/\beta \Rightarrow \frac{e^\alpha}{d^\beta} \geq 4e^{\frac{\alpha}{\beta}}$$

or

$$\alpha \geq \ln 2 + \ln n + \beta \ln d \Rightarrow \frac{e^\alpha}{d^\beta} \geq 2n$$

Therefore, $\frac{e^\alpha}{d^\beta} \geq 2e^{\frac{\alpha}{\beta}} + n$. Note that $\lfloor \frac{e^\alpha}{d^\beta} \rfloor$ is the number of vertices of degree d. In addition, G has at most n vertices of degree d, so D is continuous degree sequence and has the number of vertices at least as twice as the maximum degree.

In addition, when n is large enough, we have $\alpha = \ln 2 + \ln n + \beta \ln d$. Hence, the number of vertices N in $G_{\alpha,\beta}$ is bound as $N \leq \zeta(\beta)e^\alpha = 2\zeta(\beta)d^\beta n$, i.e. the number of vertices of $G_{\alpha,\beta}$ is polynomial bounded by the number of vertices in G_d.

□

2.4.2 Inapproximability of MIS, MVC and MDS

Theorem 2.11. *For any $\varepsilon > 0$, it is* NP*-hard to approximate Maximum Independent Set within $1 + \frac{1}{1120\zeta(\beta)3^\beta} - \varepsilon$ on simple power-law graphs.*

Proof. In this proof, we construct the simple power-law graph $G_{(\alpha,\beta)}$ based on graphic embedding technique in Theorem 2.10 from d-bounded graph G_d. Let ϕ and φ be feasible solutions of MIS on G_d and $G_{(\alpha,\beta)}$. Since $OPT(\phi) \geq n/(d+1)$ on d-bounded graphs and $N \leq 2\zeta(\beta)d^\beta n$, we further have $\mathfrak{C} = 2\zeta(\beta)d^\beta(d+1)$ from

$$OPT(\varphi) \leq N \leq 2\zeta(\beta)d^{\beta}n \leq 2\zeta(\beta)d^{\beta}(d+1)OPT(\phi)$$

According to $\epsilon = \frac{140}{139} - \varepsilon'$ for any $\epsilon' > 0$ on 3-bounded graphs, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

$$\delta > \frac{\epsilon\mathfrak{C}}{(\mathfrak{C}-1)\epsilon+1} = 1 + \frac{1}{140\mathfrak{C}-1} - \varepsilon > 1 + \frac{1}{1120\zeta(\beta)3^{\beta}} - \varepsilon$$

□

Theorem 2.12. *It is* NP*-hard to approximate Minimum Dominating Set within* $1 + \frac{1}{3120\zeta(\beta)3^{\beta}}$ *on power-law graphs.*

Proof. From the proof of Theorem 2.11, we have $\mathfrak{C} = 2\zeta(\beta)d^{\beta}(d+1)$. Then according to $\epsilon = \frac{391}{390}$ on 3-bounded graphs, we have

$$\delta > 1 + \frac{\epsilon - 1}{\mathfrak{C}} \geq 1 + \frac{1}{3120\zeta(\beta)3^{\beta}}$$

□

Theorem 2.13. *There is no* $1 + \frac{2-(2+o_c(1))\frac{\log\log c}{\log c}}{2\zeta(\beta)c^{\beta}(c+1)}$ *approximation algorithm of Minimum Vertex Cover on power-law graphs under unique games conjecture.*

Proof. Similar as the proof of Theorem 2.12, we have $\mathfrak{C} = 2\zeta(\beta)d^{\beta}(d+1)$. Then according to $\epsilon = 2 - (2 + o_d(1))\log\log d/\log d$, then the inapproximability factor can be derived from inapproximability optimal substructure framework as

$$\delta > 1 + \frac{\epsilon - 1}{\mathfrak{C}} \geq 1 + \frac{2 - (2 + o_c(1))\frac{\log\log c}{\log c}}{2\zeta(\beta)c^{\beta}(c+1)}$$

where c is the smallest d satisfying the condition in [10].

□

Theorem 2.14. *There is no* $1 + \frac{2-(2+o_c(1))\frac{\log\log c}{\log c}}{2\zeta(\beta)c^{\beta}(c+1)}$ *approximation algorithm for Minimum Positive Dominating Set on power-law graphs.*

Proof. Similar as Theorem 2.14, the proof follows from Theorem 2.8. □

2.4.3 Maximum Clique, Minimum Coloring

Lemma 4 (Ferrante *et al.* [35])**.** *Let* $G = (V, E)$ *be a simple graph with* n *vertices and* $\beta \geq 1$*. Let* $\alpha \geq \max\{4\beta, \beta \log n + \log(n+1)\}$*. Then,* $G_2 = G \setminus G_1$ *is a bipartite graph.*

Lemma 5. *Given a function* $f(x)$ ($x \in \mathbb{Z}$, $f(x) \in \mathbb{Z}^+$) *monotonously decreases, then* $\sum_x f(x) \leq \int_x f(x)$.

Corollary 2. $e^{\alpha}\sum_{i=1}^{e^{\alpha/\beta}} \left(\frac{1}{d}\right)^{\beta} < (e^{\alpha} - e^{\alpha/\beta})/(\beta - 1)$.

Theorem 2.15. *Maximum Clique cannot be approximated within* $O\left(n^{1/(\beta+1)-\epsilon}\right)$ *on large power-law graphs with* $\beta > 1$ *and* $n > 54$ *for any* $\epsilon > 0$ *unless* NP=ZPP.

Proof. In [35], the authors proved the hardness of Maximum Clique problem on power-law graphs. Here we use the same construction. According to Lemma 27, $G_2 = G \setminus G_1$ is a bipartite graph when $\alpha \geq \max\{4\beta, \beta \log n + \log(n+1)\}$ for any $\beta \geq 1$. Let ϕ be a solution on general graph G and φ be a solution on power-law graph G_2. We show the completeness and soundness.

- If $OPT(\phi) = m \Rightarrow OPT(\varphi) = m$
  If $OPT(\phi) \leq 2$ on graph G, we can solve clique problem in polynomial time by iterating the edges and their endpoints one by one. However, G is not a general graph in this case. *w.l.o.g*, assuming $OPT(\phi) > 2$, then $OPT(\varphi) = OPT(\phi) > 2$ since the maximum clique on bipartite graph is 2.

- If $OPT(\phi) \leq m/n^{1-\epsilon} \Rightarrow OPT(\varphi) < O\left(1/(N^{1/(\beta+1)-\epsilon'})\, m\right.$
  In this case, we consider the case that $4\beta < \beta \log n + \log(n+1)$, that is, $n > 54$. According to Lemma 27, let $\alpha = \beta \log n + \log(n+1)$. From Corollary 2, we have

$$N = e^{\alpha}\sum_{i=1}^{\Delta}\left(\frac{1}{i}\right)^{\beta} < \frac{e^{\alpha} - e^{\alpha/\beta}}{\beta - 1} = \frac{n^{\beta}(n+1) - n(n+1)^{1/\beta}}{\beta - 1} < \frac{2n^{\beta+1} - n}{\beta - 1}$$

  Therefore, $OPT(\varphi) = OPT(\phi) \leq m/n^{1-\epsilon} < O\left(m/\left(N^{1/(\beta+1)-\epsilon'}\right)\right)$.

□

Corollary 3. *Minimum Coloring problem cannot be approximated within* $O\left(n^{1/(\beta+1)-\epsilon}\right)$ *on large power-law graphs with* $\beta > 1$ *and* $n > 54$ *for any* $\epsilon > 0$ *unless* NP=ZPP*.*

2.5 Relationship between β and Approximation Hardness

As shown in previous sections, many hardness and inapproximability results are dependent on β. In this section, we analyze the hardness of some optimal substructure problems based on β by showing that trivial greedy algorithms can achieve constant guarantee factors for MIS and MDS.

Lemma 6. *When* $\beta > 2$*, the size of MDS of a power-law graph is greater than* Cn *where* n *is the number of vertices,* C *is some constant only dependent on* β*.*

Proof. Let $S = (v_1, v_2, \dots, v_t)$ of degrees $d_1, d_2, \dots, d_t$ be the MDS of power-law graph $G_{(\alpha,\beta)}$. Observing that the total degrees of vertices in dominating set must be at least the number of vertices outside the dominating set, we have $\sum_{i=1}^{i=t} d_i \geq |V \setminus S|$. With a given total degree, a set of vertices has minimum size when it includes the vertices of highest degrees. Since the function $\zeta(\beta - 1) = \sum_{i=1}^{\infty} \frac{1}{i^{\beta-1}}$ converges when $\beta > 2$, there exists a constant $t_0 = t_0(\beta)$ such that

$$\sum_{i=t_0}^{\Delta} i \left\lfloor \frac{e^\alpha}{i^\beta} \right\rfloor \geq \sum_{i=1}^{t_0} \left\lfloor \frac{e^\alpha}{i^\beta} \right\rfloor$$

where α is any large enough constant. Thus the size of MDS is at least

$$\sum_{i=t_0}^{\Delta} \left\lfloor \frac{e^\alpha}{i^\beta} \right\rfloor \approx \left(\zeta(\beta) - \sum_{i=1}^{t_0-1} \frac{1}{i^\beta} \right) e^\alpha \approx C|V|$$

where $C = (\zeta(\beta) - \sum_{i=1}^{t_0} \frac{1}{i^\beta})/(\zeta(\beta))$. □

Consider the greedy algorithm which selects from the vertices of the highest degree to the lowest. In the worst case, it selects all vertices with degree greater than 1 and a half of vertices with degree 1 to form a dominating set. The approximation factor of this simple algorithm is a constant.

Corollary 4. *Given a power-law graph with $\beta > 2$, the greedy algorithm that selects vertices in decreasing order of degrees provides a dominating set of size at most $\sum_{i=2}^{\Delta} \lfloor e^{\alpha}/i^{\beta} \rfloor + \frac{1}{2}e^{\alpha} \approx (\zeta(\beta) - 1/2)e^{\alpha}$. Thus the approximation ratio is $(\zeta(\beta) - \frac{1}{2})/(\zeta(\beta) - \sum_{i=1}^{t_0} 1/i^{\beta})$.*

Let us consider another maximization problem MIS, we propose a greedy algorithm Power-law-Greedy-MIS as follows. We sort the vertices in non-increasing order of degrees and start checking from the vertex of lowest degree. If the vertex is not adjacent to any selected vertex, it is selected. The set of selected vertices forms an independent set with the size at least a half the number of vertices of degree 1 which is $e^{\alpha}/2$. The size of MIS is at most a half of number of vertices. Thus, the following lemma holds.

Lemma 7. *Power-law-Greedy-MIS has factor $1/(2\zeta(\beta))$ on power-law graphs with $\beta > 1$.*

2.6 Minor *NP*-Hardness on Simple Power-Law Graphs for $\beta < 1$

In the section, we show some minor *NP*-hardness of optimal substructure problems on simple power-law graphs for small $\beta < 1$.

Definition 18 (Eligible Sequences)**.** *A sequence of integers $S = \langle s_1, \ldots, s_n \rangle$ is eligible if $s_1 \geq s_2 \geq \ldots \geq s_n$ and $f_S(k) \geq 0$ for all $k \in [n]$, where*

$$f_S(k) = k(k-1) + \sum_{i=k+1}^{n} min\{k, s_i\} - \sum_{i=1}^{k} s_i$$

Erdős and Gallai [31] showed that an integer sequence is graphic - d-degree sequence of an graph, if and only if it is eligible and the total of all elements is even. Then Havel and Hakimi [16] gave an algorithm to construct a simple graph from a degree sequence. We now prove the following eligible embedding technique based on this result.

Theorem 2.16 (Eligible Embedding Technique)**.** *Given an undirected simple graph $G = (V, E)$, $0 < \beta < 1$, there exists polynomial time algorithm to construct a power-law graph $G' = (V', E')$ of exponential factor β such that G is a set of maximal components of G'.*

Proof. To construct G', we choose $\alpha = max\{\beta \ln(n-1) + \ln(n+2), 3\ln 2\}$. Then $\lfloor e^{\alpha}/((n-1)^{\beta})\rfloor > n+2$, i.e. there are at least 2 vertices of degree d in $G' \setminus G$ if there are a least 2 vertices of degree d in G'. According to the definition, the total degrees of all vertices in G' and G are even. Therefore, the lemma will follow if we prove that the degree sequence D of $G' \setminus G$ is eligible.

In D, the maximum degree is $\lfloor e^{\alpha/\beta}\rfloor$. There is only one vertex of degree i if $1 \leq e^{\alpha}/i^{\beta} < 2$, i.e. $e^{\alpha/\beta} \geq i > (e^{\alpha}/2)^{1/\beta}$.

Let us consider $f_D(k)$ in two cases:

Case 1: $k \leq \lfloor e^{\alpha/\beta}/2\rfloor$

$$\begin{aligned} & f_D(k) = k(k-1) + \sum_{i=k+1}^{n} min\{k, d_i\} - \sum_{i=1}^{k} d_i \\ > \;& k(k-1) + \sum_{i=k}^{T-k} k + \sum_{i=B}^{k-1} i + \sum_{i=1}^{B-1} 2 - \sum_{i=1}^{k} (T-k+1) \\ = \;& k(T-k) + (k-B)(k-1+B)/2 + B(B-1) - k(2T-k+1)/2 \\ = \;& (B^2-B)/2 - k \end{aligned}$$

where $T = \lfloor e^{\alpha/\beta}\rfloor$ and $B = \lfloor (e^{\alpha}/2)^{1/\beta}\rfloor + 1$. Note that $\alpha/\beta > \ln 2\,(2/\beta + 1)$ since $\alpha > 3\ln 2$ and $0 < \beta < 1$. Hence $\left(\lfloor (e^{\alpha}/2)^{1/\beta}\rfloor + 1\right)\left(\lfloor (e^{\alpha}/2)^{1/\beta}\rfloor\right) > \lfloor e^{\alpha/\beta}\rfloor \geq 2k$, that is, $f_D(k) > 0$.

Case 2: $k > \lfloor e^{\alpha/\beta}/2\rfloor$

$$f_D(k+1) \geq f_D(k) + 2k - 2d_{k+1} \geq f_D(k) \geq ... \geq f_D(\lfloor e^{\alpha/\beta}/2\rfloor) > 0$$

□

Corollary 5. *An optimal substructure problem is also* NP*-hard on power-law graphs for all* $0 < \beta < 1$ *if it is* NP*-hard on simple general graphs.*

Proof. According to Theorem 2.16, we can embed an undirected graph $G = (V, E)$ into a power-law graph G' of β lying in $(0, 1)$ and of vertices polynomial time in the size of G. Since the optimization problem has optimal substructure property and G is a set of maximal connected components of G', its optimum solution for the graph G can be computed easily from an optimal solution for G'. This completes the proof of *NP*-hardness. □

2.7 Approximation Algorithms

As the computational hardness and inapproximability results of classic optimization problems have been shown in the previous sections, the design of approximation algorithm is still of great interest but remains open. In this section, we focus on addressing the following questions: Can the property of power-law degree distribution help us to design an effective algorithm framework for NP-hard optimization problems? How can we provide a theoretical framework for analyzing approximation ratios of these problems using this power-law degree property? Will these approximation ratios change dramatically for different exponential factors β, i.e. in power-law graphs with different densities?

We propose an algorithm framework, called Low-Degree Percolation (LDP) framework, to solve the optimization problems in power-law networks, including MIS, MDS, and MVC problems. The idea of LDP framework to percolate the graph starting from a great number of low-degree nodes in a power-law graph, allows us to develop a theoretical framework, which can be used to analysis the approximation ratios via probability theory. In particular, we apply this theoretical framework to show the approximation ratios for these problems on two well-known random power-law models in [2, 21]. At last, numerical analysis of our proposed approaches not only validates our theoretical analysis but also illustrates the effectiveness of our approaches in practice.

2.7.1 Low-Degree Percolation (LDP) Algorithm Framework

In this section, we proposed an algorithm framework to solve optimization problems by taking advantage of the degree sequence property in power-law graphs. As one can see, the most fundamental property of power-law graphs are that they contain a great number of low-degree nodes, while only a small number of high-degree nodes. Therefore, the idea of our proposed Low-Degree Percolation (LDP) algorithm framework is to sort the nodes by their degree and percolate the graph from the nodes of lowest degree. The process continues in residual graph iteratively until no more nodes, which are surely in optimal solution, can be detected. At last, we apply existing approximation approaches to detect the solution in the remaining graph.

For MDS and MVC problems, as shown in Algorithm 4, since the node incident to a node of degree 1 certainly belongs to an optimal solution, we percolate the graph by adding all the neighbors of nodes with degree 1 in each iteration. Until no more nodes of degree 1 exists in residual graph, we apply existing approximation algorithm in [83] for MDS (or [52] for MVC) to obtain the solution in this residual graph.

Algorithm 4: LDP Algorithm for MDS/MVC Problems

Input : Power-law graph G
Output: MDS (or MVC) S

```
while ∃ Nodes of degree 1 do
    foreach Node v of degree 1 do
        Add its neighbor N(v) into S;
        Remove v from G;
    end
    Remove all nodes incident to S from graph G;
end
Determine the leftover MDS (or MVC) in G using existing approximation algorithm
in [83] (or [52]) and add them into S;
return S;
```

On the other hand, Algorithm 5 shows the algorithm for MIS. In this case, the nodes of degree 1 will belong to the optimal solution, and in the meanwhile, it is certain that their neighbors cannot be in optimal solution any more. Therefore, in order to obtain

MIS, we select all nodes of degree 1 into the solution in each iteration. At last, we apply the approximation algorithm in [40] to obtain the MIS in the remaining graph.

Algorithm 5: LDP Algorithm for MIS Problem

Input : Power-law graph G
Output: MIS S

```
while ∃ Nodes of degree 1 do
    foreach Node v of degree 1 do
        Add v into S;
        Remove v and all its neighbors N(v) from G;
    end
end
Determine the leftover MIS in G using existing approximation algorithm in [40]
and add them into S;
return S;
```

Here, we note that in a special case that two nodes of degree 1 are connected, the optimal solution of MDS (or MVC, MIS) contains either one of them.

2.7.2 Approximation Ratio Analysis

In this section, we show the approximation ratio analysis of LDP Algorithms in both structural and expected random power-law networks. To do this, we first provide a theoretical framework, using LDP algorithm, to analyze the approximation ratio based on the probability that a node does not connect to any node of degree 1. Then, this framework is applied to show the ratio of optimization problems in two different models.

2.7.2.1 Theoretical framework

In this theoretical framework, as the connected component of size 2 is trivial, we mainly focus on the ratio analysis in the rest part of power-law graphs. To begin with, we first provide a formal proof of the following Lemma 8 (Similar argument for Corollary 6), which has been briefly discussed the LDP algorithms.

Lemma 8. *In the optimal solution to MDS and MVC, if we do not consider the case of connected components with size 2, there do not exist any nodes of degree 1 and all nodes incident to at least one node of degree 1 are selected.*

Proof. In the proof, let u be a node of degree 1 incident to another v of arbitrary degree larger than 1, we consider several cases: (1) If neither u and v is selected in optimal solution, no neighbor is select for u and u is not selected as well, this leads to an infeasible solution; (2) If both u and v are selected, it is easy to see that the solution is no more optimal; (3) If u is selected instead of v, we have to select a set of nodes to satisfy v if v has degree no less than 2; (4) If v is selected instead of u, both u and v are already satisfied, which means the size of the solution less than the size in a solution containing u. According to these observations, the proof is complete. □

Corollary 6. *In the optimal solution to MIS, if we do not consider the case of connected components with size 2, all nodes of degree 1 and all nodes incident to at least one node of degree 1 are selected.*

Next, we define $\mu(\alpha, \beta, i)$ to be the probability that a node v of degree i not incident to any nodes of degree 1 in a power-law graph $G_{(\alpha,\beta)}$. Our purpose is to analyze the approximation ratio based on $\mu(\alpha, \beta, i)$ in this graph $G_{(\alpha,\beta)}$.

Let X_i^u be a random variable that a node u of degree i does not connect to any nodes of degree 1. Then, we have

$$X_i^u = \begin{cases} 1, & u \in D_1 \\ 0, & u \notin D_1 \end{cases}$$

where D_1 is a set of nodes incident to at least one node of degree 1. Note that for all nodes of the same degree, they have the same random variables. For simplicity, we define X_i to be a random variable that some node of degree i. Therefore, we have the expected value of node u not incident to any nodes of degree 1 as

$$E(X_i) = \mu(\alpha, \beta, i)$$

Since the number of nodes of degree i is equal to e^α / i^β, by letting $\Delta = e^{\alpha/\beta}$ and $X = \sum_{i=2}^{\Delta} \frac{e^\alpha}{i^\beta} X_i$, we have the following lemma:

Lemma 9. *The expected number of nodes of degree no less than 2 not incident to any nodes of degree 1 is*

$$\sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}} \mu(\alpha, \beta, i)$$

Proof. The expected number of nodes not incident to any nodes of degree 1 is the sum of all nodes of degree no less than 2, i.e. $X = \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}} X_i$. Then we have

$$E(X) = \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}} E(X_i) = \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}} \mu(\alpha, \beta, i)$$

□

Lemma 10. *The variance of X is upper bounded by*

$$e^{2\alpha} \sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha, \beta, i)\chi(\alpha, \beta, j)}}{(ij)^{\beta}}$$

where $\chi(\alpha, \beta, i) = \mu(\alpha, \beta, i)(1 - \mu(\alpha, \beta, i))$.

Proof. For a random variable corresponds to a node of degree i not incident to any nodes of degree 1, the variance is

$$Var(X_i) = \Big(1 - \frac{i}{\zeta(\beta - 1)}\Big)\Big(1 - \Big(1 - \frac{i}{\zeta(\beta - 1)}\Big)\Big) = \mu(\alpha, \beta, i)(1 - \mu(\alpha, \beta, i))$$

For any two variables correspond to two nodes of degree i and j not incident to any nodes of degree 1, according to *Cauchy-Schwarz Inequality*, we have

$$|Cov(X_i, X_j)| \leq \sqrt{Var(X_i) Var(X_j)} = \sqrt{\chi(\alpha, \beta, i)\chi(\alpha, \beta, j)}$$

Then, we sum them up and obtain

$$\begin{aligned} & \sum_{X_i, X_j} |Cov(X_i, X_j)| \leq \sum_{X_i, X_j} \sqrt{Var(X_i) Var(X_j)} \\ \leq \ & \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}} \Big(\sum_{j=2}^{\Delta} \frac{e^{\alpha}}{j^{\beta}} \sqrt{\chi(\alpha, \beta, i)\chi(\alpha, \beta, j)} \Big) = e^{2\alpha} \sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha, \beta, i)\chi(\alpha, \beta, j)}}{(ij)^{\beta}} \end{aligned}$$

Therefore, we have the variance of X to be

$$Var(X) \quad = \sum_{X_i,X_j} |Cov(X_i, X_j)| \leq e^{2\alpha} \sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha,\beta,i)\chi(\alpha,\beta,j)}}{(ij)^{\beta}}$$

□

Lemma 11. *The number of nodes of degree no less than 2 which is not incident to any nodes of degree 1 is larger than* $\lambda \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}}$ *with probability at most*

$$\frac{1}{\frac{\left(\sum_{i=2}^{\Delta} \frac{1}{i^{\beta}} \left(\lambda - \mu(\alpha,\beta,i)\right)\right)^2}{\sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha,\beta,i)\chi(\alpha,\beta,j)}}{(ij)^{\beta}}} + 1}$$

Proof. Let $\phi = \lambda \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}}$, according to *One-Sided Chebyshev Inequality*,

$$\begin{aligned} & Pr[X \geq \phi] = Pr\left[X - E(X) \geq \frac{\phi - E(X)}{\sqrt{Var(X)}} \sqrt{Var(X)}\right] \\ \leq \quad & \frac{1}{\frac{(\phi - E(X))^2}{Var(X)} + 1} \leq \frac{1}{\frac{\left(\sum_{i=2}^{\Delta} \frac{1}{i^{\beta}} \left(\lambda - \mu(\alpha,\beta,i)\right)\right)^2}{\sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha,\beta,i)\chi(\alpha,\beta,j)}}{(ij)^{\beta}}} + 1} \end{aligned}$$

□

For simplicity, we define the following p_{λ} and obtain the Corollary 7.

$$p_{\lambda} = \frac{1}{\frac{\left(\sum_{i=2}^{\Delta} \frac{1}{i^{\beta}} \left(\lambda - \mu(\alpha,\beta,i)\right)\right)^2}{\sum_{i=2}^{\Delta} \sum_{j=2}^{\Delta} \frac{\sqrt{\chi(\alpha,\beta,i)\chi(\alpha,\beta,j)}}{(ij)^{\beta}}} + 1}$$

Corollary 7. *The number of nodes of degree no less than 2 incident to at least one node of degree 1 is at least* $(1 - \lambda) \sum_{i=2}^{\Delta} \frac{e^{\alpha}}{i^{\beta}}$ *with probability at least* $1 - p_{\lambda}$.

Then, based on Lemma 8, we derive the following approximation ratios of MDS and MVC in a power-law graph $G_{(\alpha,\beta)}$:

Theorem 2.17 (Main Theorem (MDS&MVC)). *In a power-law graph $G_{(\alpha,\beta)}$, by using Algorithm 4, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least $1 - p_\lambda$, where Ψ is the approximation ratio of MDS (or MVC) in Algorithm [83] (or [52]) w.r.t. a graph of size at most $e^\alpha \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$.

Proof. Let ℓ be the number of nodes incident to degree 1 in some power-law graph $G_{(\alpha,\beta)}$. We have the approximation ratio as

$$\frac{\ell + \Psi OPT}{\ell + OPT} \leq \frac{\ell + \Psi(\sum_{i=2}^{\Delta} \frac{1}{i^\beta} - \ell)}{\ell + \sum_{i=2}^{\Delta} \frac{1}{i^\beta} - \ell}$$

According to Corollary 7, we have $\ell \geq \sum_{i=2}^{\Delta} \frac{1}{i^\beta} - \lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$ with probability at least $1 - p_\lambda$. The proof is complete. □

In terms of MIS, we have the approximation ratio as follows:

Theorem 2.18 (Main Theorem (MIS)). *In a power-law graph $G_{(\alpha,\beta)}$, by using Algorithm 5, MIS can be approximated into*

$$\frac{N + e^\alpha \left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}{N + \frac{1}{\Psi} e^\alpha \left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}$$

with probability at least $1 - p_\lambda$, where N is the number of nodes with degree 1, Ψ is the approximation ratio of MIS in Algorithm [40] w.r.t. a graph of size at most $e^\alpha \sum_{i=2}^{\Delta} \frac{1}{i^\beta}$.

The proof is omitted due to its similarity of the proof in Theorem 2.17.

Next, we focus on applying this framework onto PLRG model and analyzing the approximation ratios.

2.7.2.2 Power-law random graph

In PLRG graph, the straightforward computation of $\mu_{\text{PLRG}}(\alpha, \beta, i)$ is intractable due to the difficulty to calculate all possible combinations. To this end, we consider each case that there are particular number of connected components of size 2 in PLRG. At

last, the approximation factors can be derived from the law of total probability. In the rest of this subsection, we show the probability to have τ connected component of size 2 in a PLRG graph and each $\mu^{\tau}_{\mathsf{PLRG}}(\alpha, \beta, i)$ respectively, and apply them to obtain the approximation ratios.

Lemma 12. *The probability $Pr[C_2 = \tau]$ that there are τ connected components of size 2 in a PLRG graph is*

$$\frac{\binom{w}{2\tau}(2\tau)!!\binom{N-w}{w-2\tau}(w-2\tau)!}{N!!/(N-2w-1+2\tau)!!}$$

where $N = e^{\alpha}\zeta(\beta-1)$, $w = e^{\alpha}$ is the size of nodes of degree 1.

Proof. In order to have τ connected component of size 2, 2τ mini-nodes are selected first from all w nodes of degree 1. Moreover, there are $(2\tau-1)!!$ possibilities to match these 2τ mini-nodes. Since the number of perfect matching $f(n)$ for n mini-nodes is $(n-1)!!$, the probability can be calculated by simplifying the following equation.

$$Pr[C_2 = \tau] = \frac{\binom{w}{2\tau}(2\tau)!!\binom{N-w}{w-2\tau}(w-2\tau)!f(N-2w+2\tau)}{f(N)}$$

□

Lemma 13. *In a PLRG graph G, if there are τ connected component of size 2, the probability that a node v of degree i not incident to any nodes of degree 1 is*

$$\mu^{\tau}_{PLRG}(\alpha, \beta, i) = \begin{cases} \prod_{k=0}^{w-1} \frac{N^{\tau}-i-w^{\tau}-k}{N^{\tau}-w^{\tau}-k}, & \text{If } N^{\tau}-i-w^{\tau} > w^{\tau}; \\ 0, & \text{otherwise.} \end{cases}$$

where $N^{\tau} = e^{\alpha}\zeta(\beta-1) - 2\tau$, $w^{\tau} = e^{\alpha} - 2\tau$ is the size of nodes of degree 1.

Proof. Let D_1 be a set of nodes incident to at least one node of degree 1. Consider that the whole mini-nodes are composed of three subsets, i.e., i nodes correspondent to v, w^{τ} nodes correspondent to all nodes of degree 1 and all leftover nodes, which is referred to as N_i and N_w and $N^{\tau} \setminus \{N_i \cup N_w\}$ respectively. When $N^{\tau} - i - w^{\tau} < w^{\tau}$, there are not enough mini-nodes to match all nodes of degree 1, the probability that $v \notin D_1$ is 0. Otherwise, in order for $v \notin D_1$, we have to select the nodes incident to all nodes in N_w

from $N \setminus \{N_i \cup N_w\}$.

$$Pr[v \notin D_1] = \frac{\binom{N^\tau - i - w^\tau}{w^\tau} w^\tau! f(N^\tau - 2w^\tau)}{\binom{N^\tau - w^\tau}{w^\tau} w^\tau! f(N^\tau - 2w^\tau)} = \prod_{k=0}^{w^\tau - 1} \frac{N^\tau - i - w^\tau - k}{N^\tau - w^\tau - k}$$

where $f(n) = (n-1)!!$, representing the number of perfect matching for n nodes. □

Theorem 2.19. *In a PLRG graph G, by using Algorithm 4, MDS and MVC can be approximated into*

$$1 + (\Psi - 1)\lambda$$

with probability at least

$$\prod_{\tau=0}^{\lfloor e^\alpha/2 \rfloor} Pr[C_2 = \tau](1 - p_\lambda^\tau)$$

where $p_\lambda^\tau = \frac{1}{\frac{(\lambda^\tau - \mu(\alpha,\beta,2))^2}{\chi(\alpha,\beta,2)} + 1}$ *in which* $\lambda^\tau = \lambda + \frac{\tau}{\sum_{i=2}^{\Delta} \frac{1}{i^\beta}}$.

Proof. Consider one case that there are τ connected components in the power-law graph. Thus, according to Theorem 2.17, the probability that the approximation ratio is smaller than $1+(\Psi-1)\lambda$ is $1-p_\lambda^\tau$ for $p_\lambda^\tau = \frac{1}{\frac{(\lambda^\tau - \mu(\alpha,\beta,2))^2}{\chi(\alpha,\beta,2)} + 1}$ where $\lambda^\tau = \lambda + \frac{\tau}{\sum_{i=2}^{\Delta} \frac{1}{i^\beta}}$. Therefore, according to the law of total probability, the theorem follows by taking into account all τ, which ranges from 0 up to $\lfloor e^\alpha/2 \rfloor$. □

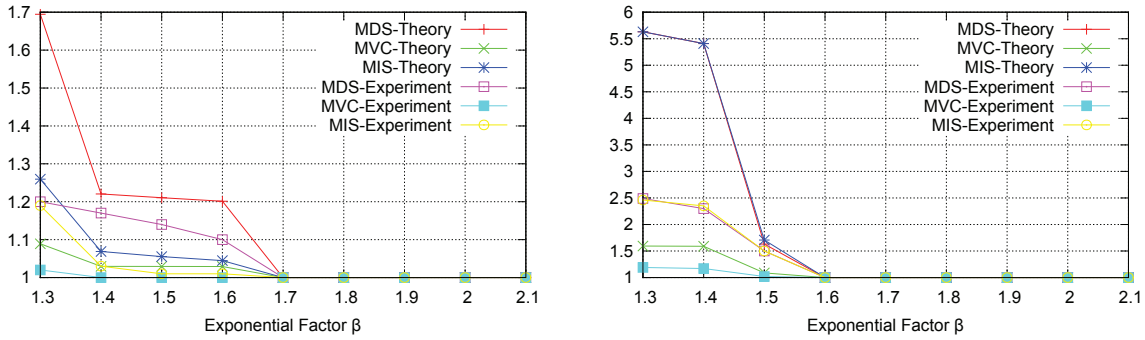
For MIS problem, the approximation ratio can be obtained as

$$\frac{N + e^\alpha \left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}{N + \frac{1}{\Psi} e^\alpha \left(\lambda \sum_{i=2}^{\Delta} \frac{1}{i^\beta}\right)}$$

with probability at least $\prod_{\tau=0}^{\lfloor e^\alpha/2 \rfloor} Pr[C_2 = \tau](1 - p_\lambda^\tau)$, where $p_\lambda^{\tau\prime} = \frac{1}{\frac{(\lambda^{\tau\prime} - \mu(\alpha,\beta,2))^2}{\chi(\alpha,\beta,2)} + 1}$ in which $\lambda^{\tau\prime} = \lambda + \frac{\tau}{\sum_{i=2}^{\Delta} \frac{1}{i^\beta}}$.

Numerical Analysis

Fig. 2-3 illustrates the performance of our LDP algorithms in random power-law graphs, along with the relation between different β and the corresponding approximation ratios, from both theoretical and practical perspectives.



A ERPL Model

B SRPL Model

Figure 2-3. Numerical results of our LDP algorithms on different β ($\alpha = 5$): (1) Theoretical results shows the approximation ratios with probability at least $1 - o(1)$. As one can see, our LDP algorithms can obtain the optimal solution for all these problems after β gets larger than 1.6 and 1.7 in ERPL and SRPL respectively, which covers the range of β in most real-world networks [18]. For the other smaller exponential factors β, we can see that the approximation ratios are a little bit higher, especially up to 5 for MDS and MIS problems for SRPL model. However, the probabilities that these two problems can obtain the approximation ratios less than 1.5 using LDP algorithms are at least 0.95 (only a little bit lower than $1 - o(1)$). (2) Experimental results further reveals that our LDP algorithms can achieve even better solutions than theoretical bounds. (We tests on 100 cases and choose the average.) As illustrated in Fig. 2-3, the approximation ratios of all MDS,MVC,MIS problems is no larger than 1.2 and 2.5 even when $\beta = 1.3$ in ERPL and SRPL models respectively.

2.8 Related Works

Many experimental results on random power-law graphs give us a belief that the problems might be much easier to solve on power-law graphs. Eubank *et al.* [32] showed that a simple greedy algorithm leads to a $1 + o(1)$ approximation factor on MINIMUM DOMINATING SET (MDS) and MINIMUM VERTEX COVER (MVC) on power-law graphs (without any formal proof) although MDS and MVC has been proved *NP*-hard to be approximated within $(1 - \epsilon) \log n$ and 1.366 on general graphs respectively [28]. In [73], Gopal also claimed that there exists a polynomial time algorithm that guarantees a $1 + o(1)$ approximation of the MVC problem with probability at least $1 - o(1)$. Unfortunately, there is no such formal proof for this claim either. Furthermore,

several papers also have some theoretical guarantees for some problems on power-law graphs. Gkantsidis *et al.* [36] proved the flow through each link is at most $O(n \log^2 n)$ on power-law random graphs where the routing of $O(d_u d_v)$ units of flow between each pair of vertices u and v with degrees d_u and d_v. In [36], the authors take advantage of the property of power-law distribution by using the structural random model [2, 2] and show the theoretical upper bound with high probability $1 - o(1)$ and the corresponding experimental results. Likewise, Janson *et al.* [48] gave an algorithm that approximated MAXIMUM CLIQUE within $1 - o(1)$ on power-law graphs with high probability on the random poisson model $G(n, \alpha)$ (i.e. the number of vertices with degree at least i decreases roughly as n^{-i}). Although these results were based on experiments and various random models, they raise an interest in investigating hardness and inapproximability of optimization problems on power-law graphs.

Recently, Ferrante *et al.* [35] had an initial attempt on power-law graphs to show the *NP*-hardness of MAXIMUM CLIQUE (CLIQUE) and MINIMUM GRAPH COLORING (COLORING) ($\beta > 1$) by constructing a bipartite graph to embed a general graph into a power-law graph and *NP*-hardness of MVC, MDS and MAXIMUM INDEPENDENT SET (MIS) ($\beta > 0$) based on their optimal substructure properties.

CHAPTER 3
VULNERABILITY ASSESSMENT

In this chapter, using the well-known random power-law graph model (SRPL) in [2], we did an in-depth analysis using probability theory with respect to different kinds of threats: random failures, preferential attacks and degree-centrality attacks. Our significant conclusions are (1) A complex network can tolerate random failures if its exponential factor is less than 2.9, (2) Power-law networks are more robust under preferential node attacks and degree-centrality node attacks when they have smaller exponential factor β, and (3) In order to maintain a reliable complex system, we optimize the power-law networks by investigating on the optimal range of exponential factor β beforehand. For both communication networks and social networks, the best β is illustrated to be lying in the interval $[1.8, 2.5]$, which gives a decent explanation to the structures of real-world networks [4, 12, 34, 74]. When $\beta < 1.8$, the maintenance of network is very costly, and when $\beta > 2.5$, the network vulnerability is unpredictable due to its dependence on the specific attacking strategy. (3) When cascading failures occur, power-law networks become extremely vulnerable when the failures can be propagated more than 2 hops.

3.1 Metric

One of the most crucial question is which *measure* copes with the network vulnerability the best? There have been many studies proposing different metrics to account for the network vulnerability [3, 5, 60, 63], among which the degree of suspected nodes or edges [5], the average shortest path length [3], the global clustering coefficients [60], the available number of compromised $s - t$ flows [63], the diameters, the relative size of the largest cluster and the average size of the isolated clusters [5] appear to be the most popular and effective. Unfortunately, these mentioned measures do not seem to cast well for some particular kinds of network vulnerabilities, especially when network fragmentation is of high priority, as depicted in Figure 1.

Let us consider a simple example in Figure 3-1 illustrating a small portion of the Internet, where nodes $v_1, v_2, \ldots, v_7$ are ISPs and the rest are consumers or transmission nodes. As revealed in this figure, any successful corruptive attacks to nodes v_8 and v_{10} are sufficient to bring the whole network down to its knees with no satisfied customers. In a different attacking strategy, the removal of node v_7 or v_9, if the adversary was to use maximum degree centrality as the metric, does not appear to harm the network function because all customers are still satisfied. These removals also reduce the global clustering coefficients to 0 and increase the average shortest path to nearly 3. Besides, if the attacker uses the available number of compromised flows from v_1 to v_2, the destructions of nodes v_4 and v_7 will drop the flow to 1, and they still, unfortunately, cannot destroy the existence of the giant ISP component providing services to the (almost) whole network.

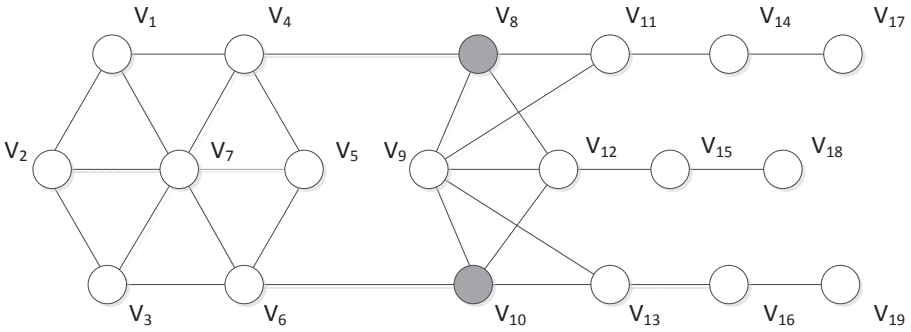


Figure 3-1. An Example of Internet: the removal of v_8 and v_{10} (grey nodes) is sufficient to destroy the function of the whole network such that only less than 40% nodes connect each other.

This example illustrates an important point that the other metrics are lack of: In order to break down the network, we need to somehow control the balance among disconnected components while ensuring the nonexistence of giant components. One possible and effective way to do so is to measure the *total pairwise connectivity* ($\mathbb{P}$), i.e. the number of connected node-pairs [27] in the network. Back to our example, a scrutiny look into the destructions of nodes v_8 and v_{10}, which we know can break

down the network function, reveals that they, indeed, reduce the network total pairwise connectivity to its greatest extent (more than 60%). This great reduction, as a result, significantly malfunctions the whole network. The measure $\mathbb{P}$ also lends itself effectively a lot of practical network applications. As we discussed above, since many large-scale networks have been shown to be power-law networks, the removal of critical nodes and links regarding this metric not only reduces the network performance but also can possibly disconnect those networks from the outside world. Another application of this metric can be found in destroying terrorist networks, e.g. to breakdown the communication between any two terrorist individuals to the greatest extent, as well as protecting the functionality in communication networks.

3.2 Threat Taxonomy and Notations

In the rest of this chapter, we focus on investigating the vulnerability of power-law networks under random failures or intentional attacks. This section consists of the following parts: (1) threat taxonomy, including random failures and intentional attacks, and (2) useful notations.

3.2.1 Threat Taxonomy

In this paper, we focus on investigating the robustness of power-law networks under random failure and two types of intentional attacks, i.e. preferential attack and degree-centrality attack.

Definition 19 (Random Failure)**.** *Each node in $G_{(\alpha,\beta)}$ fails randomly with the same probability.*

Definition 20 (Preferential Attack)**.** *Each node in $G_{(\alpha,\beta)}$ is attacked with higher probability if it has a higher degree.*

Definition 21 (Degree-Centrality Attack)**.** *The adversary only attacks the set of degree-centrality nodes in $G_{(\alpha,\beta)}$.*

Definition 22 (Random Cascading Failures)**.** *Each node in $G_{(\alpha,\beta)}$ fails randomly with the same probability, and the failures can be cascaded.*

3.2.2 Notation Explanation

With respect to each threat, we define the residual networks of the power-law network $G_{(\alpha,\beta)}$ as G_r, G_p and G_c after the occurrence of random failure, preferential attack and degree-centrality attack respectively. Their corresponding expected degree sequences are denoted as $\vec{d}_r$, $\vec{d}_p$ and $\vec{d}_c$, where the number of d_i^r, d_i^p and d_i^c are referred to as y_i^r, y_i^p and y_i^c.

In addition, we define a power-law network under certain threats to be *highly-connected* if a.s. its pairwise connectivity $\mathbb{P} = \Theta(n^2)$ and *lowly-connected* otherwise.

3.3 Preliminaries

In this section, we first present some useful results in the literature, which illustrate the important relations between the size of largest connected components and the degree sequence in random networks. Based on them, we then derive some fundamental results to evaluate the robustness of power-law networks. In this paper, the size of a connected component $S \subseteq G$ is the total number of nodes in S and the connected component S is called giant component if its size is $\Theta(n)$.

3.3.1 Previous Works

Lemma 14 (M. Molloy and B. Reed [68]). *In a random graph G with $\lambda_i n$ nodes of degree i where $\sum_{i=1}^{\Delta} \lambda_i = 1$ for the maximum degree Δ,*

$$Q = \sum_{i=1}^{n} i(i-2)\lambda_i \tag{3–1}$$

is a metric which can be applied to determine whether there is giant components in G. The giant components exist if $Q > 0$ and $\Delta < n^{1/4} - \epsilon$. Otherwise, there is a.s. no giant component if $Q < 0$ and $\Delta < n^{1/8} - \epsilon$.

Lemma 15 (F. Chung *et al.* [21]). *In a random graph G with degree sequence $\vec{d} = (d_1, d_2, \dots, d_n)$, the giant component a.s. exists if its expected average degree $\overline{d}$ is at least 1, and there is a.s. no giant component if its expected second-order average degree $\tilde{d}$ is at most 1. Furthermore, all connected components have volume (the sum*

of degrees in a connected component) at most $\sqrt{n}\log n$ *with probability at least* $1 - o(1)$ *if* $\tilde{d} < 1$*. Here the expected average degree* $\overline{d}$ *and second-order average degree* $\tilde{d}$ *are defined as*

$$\overline{d} = \frac{1}{n}\sum_{i=1}^{n} d_i, \quad \tilde{d} = \frac{\sum_{i=1}^{n} d_i^2}{\sum_{i=1}^{n} d_i} \tag{3–2}$$

where d_i *is the elements in the degree sequence.*

Corollary 8. *All connected components a.s. have sizes at most* $\frac{1}{2}\sqrt{n}\log n + 1$ *if* $\tilde{d} < 1$*.*

Proof. Consider a connected component S, the volume of S is defined as $\text{Vol}(S) = \sum_{v_i \in S} d_i$. Since there are at least $|S| - 1$ edges in a connected component of size $|S|$, we have $2(|S| - 1) \leq \text{Vol}(S) \leq \sqrt{n}\log n$. Therefore, the size of S is upper bounded by $\frac{1}{2}\sqrt{n}\log n + 1$. □

3.3.2 Robustness of Intact Power-law Networks

Theorem 3.1. *For a power-law network represented as a* (α, β) *graph* $G_{(\alpha,\beta)}$*,*

- *If* $\beta < 3.47875$*, the pairwise connectivity* $\mathbb{P}$ *is* $\Theta(n^2)$*;*
- *If* $\beta \geq 3.47875$*, the range of pairwise connectivity* $\mathbb{P}$ *is a.s. at most* $\frac{1}{2}n\left(c(\beta)n^{\frac{2}{\beta}}\log n - 1\right)$*.*

where $c(\beta) = 16/\left[\zeta(\beta)\left(2 - \frac{\zeta(\beta-2)}{\zeta(\beta-1)}\right)\right]^2$ *is a constant on any given* β*.*

To prove Theorem 3.1, we first show the relation between the largest component and our metric, the total pairwise connectivity, in the following lemma.

Lemma 16. *Suppose that the maximum size of a connected component in the graph* $G = (V, E)$ *is* ℓ*, the pairwise connectivity* $\mathbb{P}$ *is then at most* $\frac{n(\ell-1)}{2}$*.*

Proof. To prove the upper bound, we consider the worst case that the whole network consists of all connected components of size ℓ except some leftover nodes. Suppose that there are c_1 connect components of size ℓ and the number of leftover nodes is c_2, we have $n = c_1\ell + c_2$. Therefore, the pairwise connectivity $\mathbb{P}$ is

$$\mathbb{P} \leq c_1\binom{\ell}{2} + \binom{c_2}{2} \leq c_1\binom{\ell}{2} + \frac{c_2}{\ell}\binom{\ell}{2} = \frac{c_1\ell + c_2}{\ell}\binom{\ell}{2} = \frac{n(\ell-1)}{2}$$

□

Proof of Theorem 3.1:

Proof. First, according to F. Chung *et al.* [2], we can find the threshold 3.47875 of β such that $Q > 0$ when $\beta < 3.47875$ and $Q < 0$ when $\beta > 3.47875$.

When $\beta < 3.47875$, according to Lemma 14, since $Q > 0$, there exists one giant component of size $\Theta(n)$. Therefore, the pairwise connectivity $\mathbb{P}$ is $\Theta(n^2)$.

When $\beta > 3.47875$, according to Aiello *et al.* [2], a connected component S in the (α, β) graph a.s. has the size at most $c(\beta)n^{\frac{2}{\beta}} \log n$. Then the upper bound of $\mathbb{P}$ follows directly from Lemma 16. □

In the following three sections, since the power-law networks with β at least 3.47875 are lowly-connected even if they are not attacked, we will focus on exploiting the robustness of power-law networks with β less than 3.47875 under random failures, preferential attacks and degree-centrality attacks respectively.

3.4 Random Failures

In this section, we focus on the robustness of power-law networks after random failures, in which each node has the same probability p ($0 < p < 1$) to fail. The total pairwise connectivity $\mathbb{P}$ in the residual graph G_r is proven as in the following Theorem 3.6. Based on this theorem, we further investigate the good range of exponential factor β.

3.4.1 Robustness under Random Failures

Theorem 3.2. *In a residual graph G_r of $G_{(\alpha,\beta)}$ after random failures,*

- *If $\beta < \beta_p$, the expected pairwise connectivity $E(\mathbb{P})$ is a.s. $\Theta(n^2)$;*
- *If $\beta \geq \beta_p$, the pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{2}n\left(c_r(\beta)n^{\frac{2}{\beta}} \log n - 1\right)$.*

where β_p satisfies that $(1-p)\zeta(\beta_p - 2) - (2-p)\zeta(\beta_p - 1) = 0$ and

$$c_r(\beta) = 16/\left[\zeta(\beta)\left(2 - p - (1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)}\right)\right]^2$$

.

To prove Theorem 3.6, we first show the expected degree distribution in G_r as follows.

Lemma 17. *The expected degree distribution of graph G_r is*

$$E(y_i^r) = (1-p)^{i+1} \sum_{k=i}^{\Delta} \binom{k}{i} \frac{e^{\alpha}}{k^{\beta}} p^{k-i}$$

where degree i is $1 \leq i \leq \Delta$.

Proof. Let p_k^i be the probability that a node v of degree k in $G_{(\alpha,\beta)}$ has its degree to be i in G_r. When $k < i$, it is clear that $p_k^i = 0$; otherwise when $k \geq i$, v will become a node of of degree i in G_r if and only if v itself does not fail but $k - i$ of its neighbors fail. Hence, the probability p_k^i is $\binom{k}{i}(1-p)[p^{k-i}(1-p)^i]$, i.e. $\binom{k}{i}p^{k-i}(1-p)^{i+1}$.

Thus, according to the basic definition of expected value, the expected number of nodes of degree i in G_r is

$$\mathrm{E}(y_i^r) = \sum_{k=1}^{\Delta} p_k^i \frac{e^{\alpha}}{k^{\beta}} = (1-p)^{i+1} \sum_{k=i}^{\Delta} \binom{k}{i} \frac{e^{\alpha}}{k^{\beta}} p^{k-i}$$

□

Proof of Theorem 3.6:

Proof. First of all, we show that Lemma 15 cannot be applied here. Consider the expected second-order average degree $\tilde{d}_r$ of G_r, we have

$$\tilde{d} = p + (1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)}$$

It is easy to see that $\tilde{d} > 1$ for any p and β.

In an alternative way, we use Lemma 14 and branching process method to prove our theorem. *The basic idea is as follows*: according to the expected degree of G_r, we first find a threshold β_p using Lemma 14, which determines whether the total pairwise connectivity $\mathbb{P}$ of the residual network G_r is a.s. $\Theta(n^2)$ or not. If not, that is, $\beta > \beta_p$, we further use branching process method to prove that $\mathbb{P}$ in G_r is a.s. at most

$\frac{1}{2}n\left(c_r(\beta)n^{\frac{2}{\beta}}\log n - 1\right)$. First, we compute β_p for G_r as

$$Q_r = \sum_{i=1}^{\Delta} i(i-2)(1-p)^{i+1} \sum_{k=i}^{\Delta} \binom{k}{i} \frac{e^{\alpha}}{k^{\beta}} p^{k-i} \tag{3–3a}$$

$$= e^{\alpha}(1-p) \sum_{i=1}^{\Delta} \frac{1}{i^{\beta}} \sum_{j=1}^{i} j(j-2) \binom{i}{j} p^{i-j}(1-p)^{j} \tag{3–3b}$$

$$= e^{\alpha}(1-p)^2 \sum_{i=1}^{\Delta} \frac{i^2(1-p) - i(2-p)}{i^{\beta}} \tag{3–3c}$$

$$\doteq e^{\alpha}(1-p)^2 \left[(1-p)\zeta(\beta-2) - (2-p)\zeta(\beta-1)\right] \tag{3–3d}$$

where step (3–3c) follows similarly from the calculation of expected value and variance in binomial distribution.

Let us consider the case that the threshold β_p satisfies $(1-p)\zeta(\beta-2) - (2-p)\zeta(\beta-1) = 0$. When $\beta < \beta_p$, we have $Q_r > 0$. Thus, the expected pairwise connectivity $E(\mathbb{P})$ is a.s. $\Theta(n^2)$ according to Lemma 14.

Algorithm 6: Branching Process Method

```
i ← 0;
E_0 = L_0 = {v} by picking an arbitrary node v;
while |L_i| ≠ 0 do
    i ← i + 1;
    Choose an arbitrary u from L_{i-1} and expose all its neighbors N(u);
    E_i = E_{i-1} ∪ N(u);
    L_i = (L_i \ ({u}) ∪ (N(u) \ E_{i-1});
end
```

When $\beta > \beta_p$, we use the following branching process method (Algorithm 6) on G_r according to its expected degree sequence $E(y_i^r)$. In the algorithm, we define E_i and L_i as the set of exposed nodes and live nodes in iteration i respectively, where live nodes are referred to as the subset of exposed nodes whose neighbors have not been exposed. Note that $|L_i| = 0$ if and only if the entire component is exposed. For simplicity, we define random variables $\mathcal{E}_i = |E_i|$ and $\mathcal{L}_i = |L_i|$ as the number of exposed nodes and live nodes. Let $\mathcal{T}$ denote the whole number of iterations in branching process, that is, $\mathcal{T}$

also measures the size of connected component since exactly one node is exposed in each iteration. We further define an edge to be a "backedge" if it connects u and some node in E_{i-1}. We denote $D_i = |N(u)|$ and $B_i = |N(u) \cap E_{i-1}| - 1$ measuring the degree of the node exposed in iteration i and the number of "backedge". By definition, we have $\mathcal{L}_i - \mathcal{L}_{i-1} = D_i - B_i - 2$ immediately.

Then, we calculate $\mathsf{E}(D_i)$, $\mathsf{E}(B_i)$ and $\mathsf{E}(\mathcal{L}_i)$ respectively. Consider one edge in original graph $G_{(\alpha,\beta)}$. It still exists iff both endpoints are not failed, that is, the expected number of edges in G_r is $(1-p)^2 m$. Therefore,

$$\begin{aligned}
\mathsf{E}(D_i) &= \sum_{i=1}^{\Delta} i \frac{i(1-p)^{i+1} \sum_{k=i}^{\Delta} \binom{k}{i} \frac{e^{\alpha}}{k^{\beta}} p^{k-i}}{(1-p)^2 m} \\
&= \frac{1}{\zeta(\beta-1)} \sum_{i=1}^{\Delta} \frac{i^2(1-p)+ip}{i^{\beta}} \\
&\doteq (1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)} + p
\end{aligned}$$

Since $|N(u) \cap E_{i-1}| \geq 1$, we have $\mathsf{E}(B_i) \geq 0$. By substituting $\mathsf{E}(D_i)$ and $\mathsf{E}(B_i)$ into $\mathcal{L}_i - \mathcal{L}_{i-1} = D_i - B_i - 2$, we have

$$\begin{aligned}
\mathsf{E}(\mathcal{L}_i) &= \mathcal{L}_1 + \sum_{j=2}^{i} \mathsf{E}(\mathcal{L}_j - \mathcal{L}_{j-1}) \\
&= d_0 + \sum_{j=2}^{i} \mathsf{E}(D_j - B_j - 2) \\
&\leq d_0 + (i-1)\left((1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)} + p - 2\right) \\
&= d_0 - \lambda(p,\beta)(i-1)
\end{aligned}$$

where $\lambda(p,\beta) = 2 - p - (1-p)\frac{\zeta(\beta-2)}{\zeta(\beta-1)}$ and the initial node is assumed to have degree d_0.

Since $|\mathcal{L}_j - \mathcal{L}_{j-1}| \leq \Delta = e^{\frac{\alpha}{\beta}}$, according to *Azuma Martingale Inequality* [22],

$$\Pr\left[|\mathcal{L}_i - E(\mathcal{L}_i)| > \mathcal{T}\right] \leq 2e^{\frac{-\mathcal{T}^2}{2ie^{\frac{2\alpha}{\beta}}}}$$

where $i = \frac{16}{(\lambda(p,\beta))^2} e^{\frac{2\alpha}{\beta}} \log n = c_r(\beta) n^{\frac{2}{\beta}} \log n$ and $\mathcal{T} = \lambda(p,\beta) i/2$. Since we know

$$\mathsf{E}(\mathcal{L}_i) + \mathcal{T} \leq d_0 - \lambda(p,\beta)(i-1) + \frac{\lambda(p,\beta)}{2} i < 0$$

for any d_0. Therefore,

$$\begin{aligned} & \Pr\left[\mathcal{T} > \frac{16}{(\lambda(p,\beta))^2} e^{\frac{2\alpha}{\beta}} \log n\right] = \Pr[\mathcal{T} > i] \\ \leq\ & \Pr[\mathcal{L}_i > 0] \leq \Pr[\mathcal{L}_i > \mathsf{E}(\mathcal{L}_i) + \mathcal{T})] \\ \leq\ & 2e^{\frac{-\mathcal{T}^2}{2ie^{\frac{2\alpha}{\beta}}}} = \frac{2}{n^2} \end{aligned}$$

Thus, the probability that, in graph G_r, there is a non-failure node v belonging to a connected component of size larger than $c_r(\beta) n^{\frac{2}{\beta}} \log n$ is at most $n\frac{2}{n^2} = o(1)$, i.e. G_r has the largest connected component of size a.s. $c_r(\beta) n^{\frac{2}{\beta}} \log n$. Hence, the upper bound of pairwise connectivity in G_r follows from Lemma 16 directly. □

3.4.2 Good Range of β under Random Failures

According to Theorem 3.6, we exploit the good range of exponential factor β in terms of the pairwise connectivity $\mathbb{P}$ of power-law networks. By obtaining the threshold β_p from $(1-p)\zeta(\beta_p - 2) - (2-p)\zeta(\beta_p - 1) = 0$, the relation between threshold β_p and failure probability p can be revealed in the following Fig. 3-2.

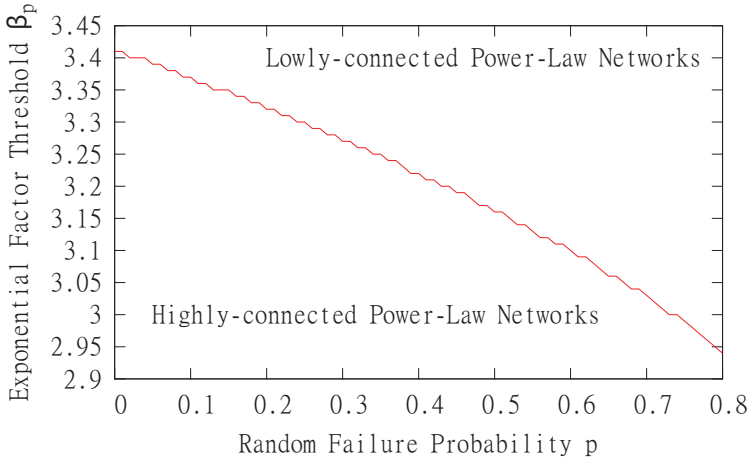


Figure 3-2. Relation between Threshold β_p and Failure Probability p

Based on Theorem 3.6, power-law networks are highly-connected when $\beta > \beta_p$ and lowly-connected otherwise. As one can see from Fig. 3-2, power-law networks of exponential factor $\beta > 2.9$ will still remain highly-connected under random failures even when the failure probability p is unrealistically 0.8. That is, we can confidently claim that power-law networks have an extremely high tolerance to random failures when its exponential factor $\beta < 2.9$.

3.5 Preferential Attacks

As power-law networks are tolerable to random failures, one will question whether it can still tolerate intentional attacks if the intruders intend more to attack "hub" nodes. In this section, we focus on the robustness of power-law networks under preferential attacks. As we defined above, in preferential attacks, each node in the network is attacked with higher probability if it has larger degree. Therefore, consider the costs to attack for intruders, we focus on the following two preferential attack schemes:

- *Interactive Preferential Attacks:* one way to control the costs to attack is to attack a node w.r.t. its degree and a new parameter β'. That is, a node of degree i is attacked with probability $1 - \frac{1}{i^{\beta'}}$;
- *Expected Preferential Attacks:* another way to control the costs to attack is based on the expected number of nodes c to attack. When the intruder decides c, ranging between 0 and $e^{\alpha}\zeta(\beta)$, a node of degree i is attacked with probability $p_i = c\frac{i}{e^{\alpha}\zeta(\beta-1)}$ since the expected number of failure nodes is equal to c, namely $\sum_i \frac{e^{\alpha}}{i^{\beta}} p_i = c$.

As one can see, in both these schemes, a node of higher degree, often referred to as a "hub", is *more preferentially* attacked, that is, it has higher probability to be attacked. By denoting their corresponding residual graphs as G_p^I and G_p^E, their total pairwise connectivity are proven in Theorem 3.3 and Theorem 3.4 respectively.

3.5.1 Interactive Preferential Attacks $\left(p_i = 1 - \frac{1}{i^{\beta'}}\right)$

Theorem 3.3. *In a residual graph G_p^I of $G_{(\alpha,\beta)}$ after interactive preferential attacks,*

- *If $\beta + \beta' < 3.47875$, the expected pairwise connectivity $E(\mathbb{P})$ is $\Theta(n^2)$;*

- *If* $\beta+\beta' \geq 3.47875$*, the pairwise connectivity* $\mathbb{P}$ *is a.s. at most* $\frac{1}{2}n\left(c(\beta)n^{\frac{2}{\beta}}\log n-1\right)$*. where* $c(\beta)=16/\left[\zeta(\beta)\left(2-\frac{\zeta(\beta-2)}{\zeta(\beta-1)}\right)\right]^2$ *is a constant on any given* β*.*

Theorem 3.3 can be proven in the same way as in Theorem 3.1 after showing the expected degree in residual graph G_p^l as in Lemma 20, which is based on the following two lemmas.

Lemma 18. *In graph* $G_{(\alpha,\beta)}$*, the probability that a node* v *of degree* i *incident to another node* u *of degree* x *is* $\frac{ix}{e^{\alpha}\zeta(\beta-1)}$*.*

Proof. Consider a node v of degree i, in the matching of mini-nodes, at least one of i mini-nodes for v connects to another one of x for node u of degree x. Thus, we have

$$\frac{\binom{i}{1}\binom{x}{1}f(N-2)}{f(N)}=\frac{ix}{N-1}=\frac{ix}{N}+O(\frac{1}{N^2})\doteq\frac{ix}{e^{\alpha}\zeta(\beta-1)}$$

where $f(n)=(n-1)!!$ representing the number of perfect matchings for N nodes and $N=e^{\alpha}\zeta(\beta-1)$ denotes the number of mini-nodes. □

Lemma 19. *For a node* v *of degree* i*, the expected number of non-failure neighbors* $E(N_p^l(i))$ *of* v *is* $i\frac{\zeta(\beta+\beta'-1)}{\zeta(\beta-1)}$*.*

Proof. According to Lemma 18, node v has probability $\frac{ix}{e^{\alpha}\zeta(\beta-1)}$ to connect to node u of degree x. Since node u of degree x has the non-failure probability $\frac{1}{x^{\beta'}}$, then we have the expected non-failure neighbor of v to be

$$\mathrm{E}(N_p^l(i))\doteq\sum_{x=1}^{\Delta}\frac{ix}{e^{\alpha}\zeta(\beta-1)}\frac{1}{x^{\beta'}}\frac{e^{\alpha}}{x^{\beta}}$$

$\doteq i\frac{\zeta(\beta+\beta'-1)}{\zeta(\beta-1)}$ *Theproofiscomplete.* □

Lemma 20. *The expected degree distribution of graph* G_p^l *is*

$$E(y_i^p)\doteq\frac{e^{\alpha}}{\left(i\frac{\zeta(\beta-1)}{\zeta(\beta+\beta'-1)}\right)^{\beta+\beta'}}$$

where $i\in\left\{\frac{\zeta(\beta+\beta'-1)}{\zeta(\beta-1)},\ldots,\Delta\frac{\zeta(\beta+\beta'-1)}{\zeta(\beta-1)}\right\}$*.*

Proof. Consider the set of nodes with degree i in G_p, they are correspondent to the nodes of degree x in the original graph $G_{(\alpha,\beta)}$. Hence, the expected unattacked nodes in this set is $\frac{e^\alpha}{x^\beta}\frac{1}{x^{\beta'}} = \frac{e^\alpha}{x^{\beta+\beta'}}$. According to Lemma 19, we know the relation between i and x is $i \doteq x\frac{\zeta(\beta+\beta'-1)}{\zeta(\beta-1)}$. Therefore, we have the expected number of nodes of degree i in G_p^I to be $\frac{e^\alpha}{\left(i\frac{\zeta(\beta-1)}{\zeta(\beta+\beta'-1)}\right)^{\beta+\beta'}}$. □

3.5.2 Expected Preferential Attacks $\left(p_i = c\frac{i}{e^\alpha\zeta(\beta-1)}\right)$

Theorem 3.4. *In a residual graph G_p^E of $G_{(\alpha,\beta)}$ after expected preferential attacks,*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $c < \min\left\{c \middle| \frac{\sum_x \frac{e^\alpha}{x^\beta}\left(1-\frac{cx}{e^\alpha\zeta(\beta-1)}\right)\left(x\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)\right)}{n-c} > 1\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4}n^{\frac{3}{2}}\log n$ if $c > \max\left\{c \middle| \left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)\frac{\zeta(\beta-2)-\frac{c\zeta(\beta-3)}{e^\alpha\zeta(\beta-1)}}{\zeta(\beta-1)-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)}} < 1\right\}$.*

To prove Theorem 3.4, we again first show the expected degree distribution in G_p^E as follows.

Lemma 21. *For a node v of degree i, the expected number of non-failure neighbors $E(N_p^E(i))$ of v is $i\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)$.*

Proof. According to Lemma 18, the node v has probability $\frac{ix}{e^\alpha\zeta(\beta-1)}$ to connect node u of degree x. Since node u of degree x has the non-failure probability $1-c\frac{x}{e^\alpha\zeta(\beta-1)}$, then we have the expected non-failure neighbor of v to be

$$\mathsf{E}(N_p(i)) \doteq \sum_{x=1}^{\Delta}\frac{ix}{e^\alpha\zeta(\beta-1)}\left(1-\frac{cx}{e^\alpha\zeta(\beta-1)}\right)\frac{e^\alpha}{x^\beta} \doteq i\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)$$

The proof is complete. □

Corollary 9. *The expected degree distribution of graph G_p^E is*

$$E(y_i^p) \doteq \frac{e^\alpha}{i^\beta}\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)^\beta\left(1-\frac{ci}{(e^\alpha\zeta(\beta-1))\left(1-\frac{c\zeta(\beta-2)}{e^\alpha\zeta(\beta-1)^2}\right)}\right)$$

where $i \in \left\{\left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right), \ldots, \Delta\left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right)\right\}$.

Proof of Theorem 3.4:

Proof. In the proof, we first calculate the expected average degree $\overline{y}_p^E$ based on Corollary 9 as

$$\overline{d}_p^E \doteq \frac{\sum_{x=1}^{\Delta} \frac{e^{\alpha}}{x^{\beta}}\left(1-\frac{cx}{e^{\alpha}\zeta(\beta-1)}\right)\left(x\left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right)\right)}{n-c}$$

and second-order average degree $\tilde{d}_p^E$ as

$$\begin{aligned}\tilde{d}_p^E &\doteq \frac{\sum_{x=1}^{\Delta} \frac{e^{\alpha}}{x^{\beta}}\left(1-\frac{cx}{e^{\alpha}\zeta(\beta-1)}\right)\left(x\left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right)\right)^2}{\sum_{x=1}^{\Delta} \frac{e^{\alpha}}{x^{\beta}}\left(1-\frac{cx}{e^{\alpha}\zeta(\beta-1)}\right)\left(x\left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right)\right)} \\ &\doteq \left(1-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)^2}\right)\frac{\zeta(\beta-2)-\frac{c\zeta(\beta-3)}{e^{\alpha}\zeta(\beta-1)}}{\zeta(\beta-1)-\frac{c\zeta(\beta-2)}{e^{\alpha}\zeta(\beta-1)}}\end{aligned}$$

According to Lemma 15 and Corollary 8, there exists one giant component if $\overline{y}_p^E > 1$ and all components have size at most $\frac{1}{2}\sqrt{n}\log n + 1$ if $\tilde{y}_p^E < 1$, then the proof follows from Lemma 16 directly. □

3.5.3 Relations between β and Expected Attacked Nodes

In interactive preferential attacks, according to Theorem 3.3, a power-law networks with exponential factor β will be lowly-connected if the intruder select a β' such that $\beta + \beta' \geq 3.47875$. Since a node of degree i is attacked with probability $1 - \frac{1}{i^{\beta'}}$ in this scheme, this node can survive with probability $\frac{1}{i^{\beta'}}$. Therefore, we have the expected number of survived nodes as

$$\sum_i \frac{e^{\alpha}}{i^{\beta}}\frac{1}{i^{\beta'}} \doteq e^{\alpha}\zeta(\beta+\beta')$$

that is, the expected percentage of attacked nodes can be obtained by calculating $1 - \frac{\zeta(\beta+\beta')}{\zeta(\beta)}$.

Fig. 3-3 reports the relation between β and expected attacked nodes under iterative preferential attacks. We observed that the expected number of attacked nodes

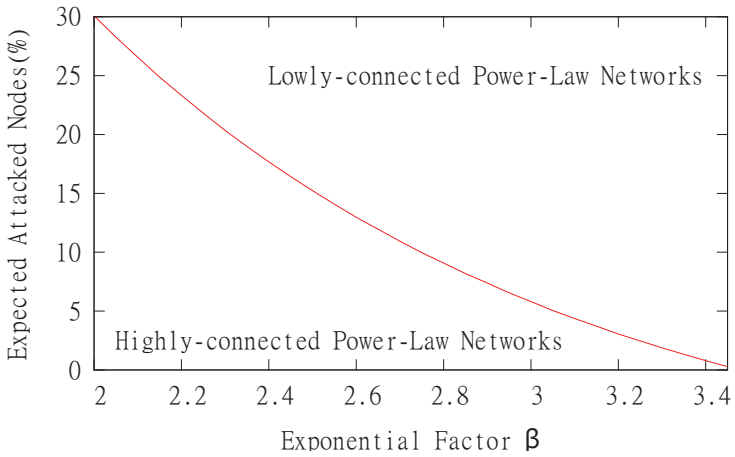


Figure 3-3. Relation between β and Attacked Nodes under Iterative Preferential Attacks

decreases sharply with the increase of β. Clearly, smaller β leads to a more robust power-law network.

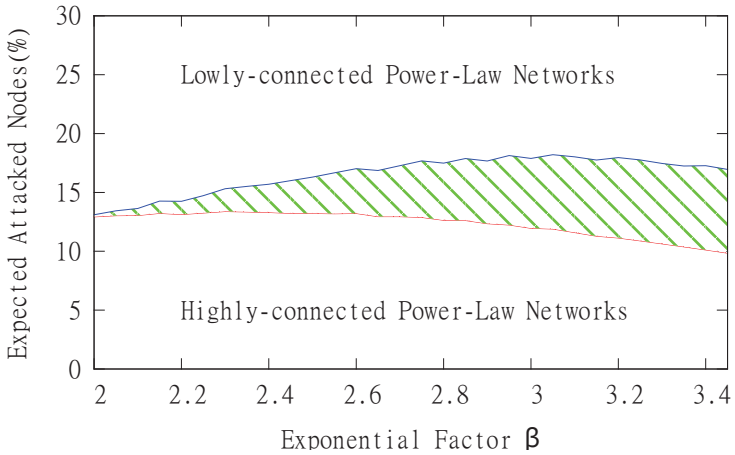


Figure 3-4. Relation between β and Attacked Nodes under Expected Preferential Attacks

In expected preferential attacks, again Fig. 3-4 reveals the smaller β the better. According to Theorem 3.4, except of uncertain areas (shadow areas), we can see that the percentage of attacked nodes (under the red line) reduces when β increases.

3.6 Degree-Centrality Attacks

As power-law networks is quite vulnerable under preferential attacks, their toleration to the deterministic intentional attacks attracts more attentions. Also, one can also

question whether it is still true under deterministic intentional attacks that power-law networks with smaller β can better maintain their functionalities. In this section, we consider the degree-centrality attack, in which the intruders intentionally attack the "hubs", the set of nodes of highest degrees. When all nodes of degree larger than x_0 are attacked simultaneously, we have the following Theorem 3.5.

3.6.1 Robustness under Degree-Centrality Attacks

Theorem 3.5. *In a residual graph G_c of $G_{(\alpha,\beta)}$ after degree-centrality attacks,*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $x_0 > \min\left\{x_0 \middle| \frac{1}{\zeta(\beta-1)} \frac{\left(\sum_{x=1}^{x_0} \frac{1}{x^{\beta-1}}\right)^2}{\sum_{x=1}^{x_0} \frac{1}{x^{\beta}}} > 1\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4} n^{\frac{3}{2}} \log n$ if $x_0 < \max\left\{x_0 \middle| \frac{1}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-2}} < 1\right\}$.*

To prove Theorem 3.5, we again first show the expected degree distribution in G_c as follows.

Lemma 22. *For a node v of degree i in original graph $G_{(\alpha,\beta)}$, the expected number of neighbors of degree larger than x_0 is $\frac{i}{\zeta(\beta-1)} \sum_{i=x_0+1}^{\Delta} \frac{1}{i^{\beta-1}}$.*

Proof. According to Lemma 19, the probability that a node v of degree i incident to a node u of degree x is $\frac{ix}{e^{\alpha}\zeta(\beta-1)}$. Therefore, we have the expected number of neighbors of degree larger than x_0 to be

$$\mathsf{E}(N_c(i)) \doteq \sum_{x=x_0+1}^{\Delta} \frac{ix}{e^{\alpha}\zeta(\beta-1)} \frac{e^{\alpha}}{x^{\beta}} = \frac{i}{\zeta(\beta-1)} \sum_{x=x_0+1}^{\Delta} \frac{1}{x^{\beta-1}}$$

□

Corollary 10. *The expected degree sequence in G_c is*

$$E(y_i^c) \doteq \frac{e^{\alpha}}{i^{\beta}} \left(\frac{1}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-1}}\right)^{\beta}$$

where $i \in \left\{\frac{1}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-1}}, \cdots, \frac{x_0}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-1}}\right\}$.

Proof of Theorem 3.5:

Proof. With the expected average degree $\overline{y}_c$ as

$$\overline{d}_c = \frac{\sum_{x=1}^{x_0} \frac{e^\alpha}{x^\beta} x \left(\frac{1}{\zeta(\beta-1)} \sum_{i=1}^{x_0} \frac{1}{i^{\beta-1}} \right)}{n - \sum_{i=x_0+1}^{\Delta} \frac{e^\alpha}{i^\beta}} = \frac{1}{\zeta(\beta-1)} \frac{\left(\sum_{x=1}^{x_0} \frac{1}{x^{\beta-1}} \right)^2}{\sum_{x=1}^{x_0} \frac{1}{x^\beta}}$$

and second-order average degree $\tilde{y}_c$ as

$$\tilde{d}_c = \frac{\sum_{x=1}^{x_0} \frac{e^\alpha}{x^\beta} \left[x \left(\frac{1}{\zeta(\beta-1)} \sum_{i=1}^{x_0} \frac{1}{i^{\beta-1}} \right) \right]^2}{\sum_{x=1}^{x_0} \frac{e^\alpha}{x^\beta} x \left(\frac{1}{\zeta(\beta-1)} \sum_{i=1}^{x_0} \frac{1}{i^{\beta-1}} \right)} = \frac{1}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-2}}$$

The rest of proof is the same as Theorem 3.4. □

3.6.2 Relations between β and Attacked Nodes

Fig. 3-5 illustrates the relations between β and attacked nodes under degree-centrality attacks based on Theorem 3.5. On the one hand, it is similar to expected preferential attacks that the percentage of attacked nodes (under the red line) reduces when β increases except of uncertain areas (shadow areas). On the other hand, under degree-centrality attacks, the intruder only needs to attack roughly 8% less number of nodes to lower down the pairwise connectivity of power-law networks than under expected preferential attacks.

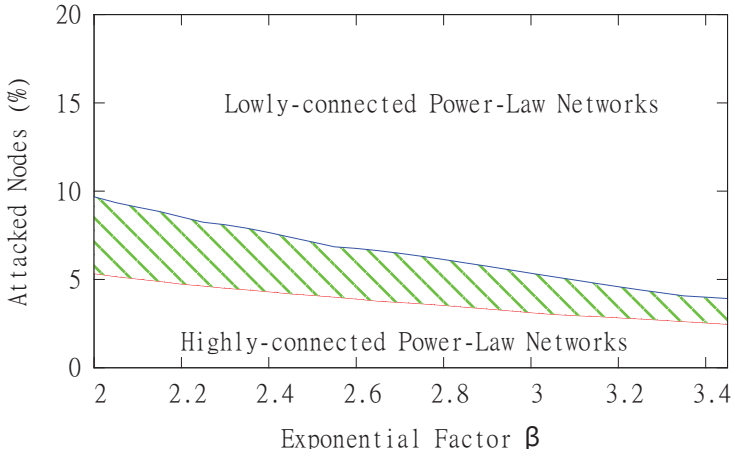


Figure 3-5. Relation between β and Attacked Nodes under Degree-Centrality Attacks

3.7 Random Cascading Failures

More importantly, the failures will lead to a much more devastating consequence especially when *failures are cascaded*, i.e., these failed nodes can cause the overload and failure of their nearby elements in the system because of the load shifting.

Let us consider an example in Figure 3-6 illustrating a small portion of the power grid, where nodes $v_1, v_2, \ldots, v_7$ are generators, $v_8, v_9, \ldots, v_{16}$ are transmitters and v_{17}, v_{18}, v_{19} are customers. Each node has load equal to its degree and capacity equal to twice its degree. As revealed in this figure, any successful corruptive attacks to nodes v_8 and v_{10} can affect the power supply from generators or transmitters instantly, while customers are still able to utilize the electricity until the left electricity in demand centers is used up. However, when failures are cascaded, all transmitters can fail sequentially (gradual color changes), i.e., $v_8, v_{10} \Rightarrow v_{12} \Rightarrow v_9, v_{15} \Rightarrow v_{11}, v_{13} \Rightarrow v_{14}, v_{16}$, leading to no power supply to customers instantly. Therefore, in order to continuously maintain the normal network functions, it is of great importance to assess the network vulnerability in the present of cascading failures, beforehand.

In this section, taking into account cascading failures, we analyze the network vulnerability via two main thrusts, complex network structure analysis and optimal detection of most vulnerable nodes, based on the recently proposed effective metric, *total pairwise connectivity* [27, 77, 78]. By measuring the connected node-pairs in residual networks (a pair of nodes are connected when there is a functional path between them), the minimum of total pairwise connectivity can control the balance among disconnected components and further ensure the nonexistence of giant components, leading to the destruction of network functionality.

3.7.1 Cascading Failure Model

In this paper, we use one of the most well-accepted models proposed in [85], in which each node u in the network has a threshold $\theta_u \in [0, 1]$, typically drawn from some probability distribution. Starting with an initial set of failure nodes F_0, called

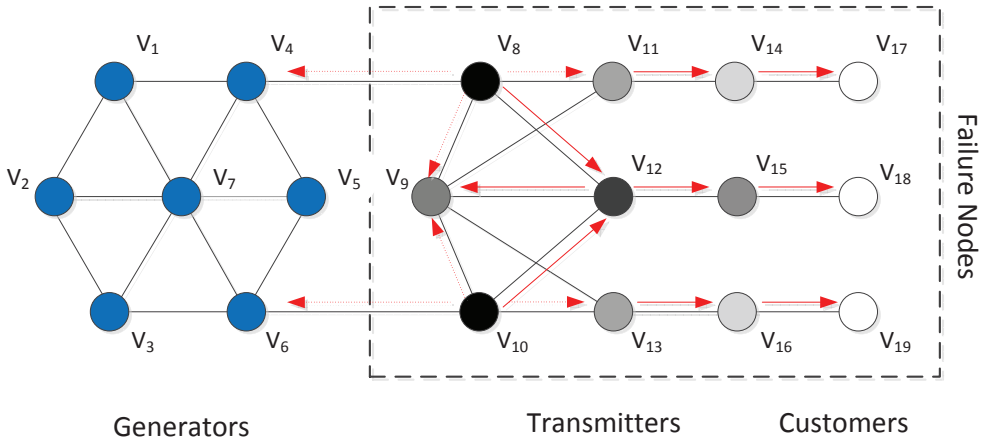


Figure 3-6. Each node in this power grid has load equal to its degree, capacity equal to twice its degree and each red arrow says the shifting of 2 unit load. The solid red arrows stand for the direct failure caused by the cascades and the dotted ones mean the load shifting to the neighbor which is not failed directly. The overload and failure of v_8 and v_{10} can only cause the disconnection from generators and transmitters, yet the power can be still supplied to customers from demand centers. However, when failure cascades, it leads to the breakdown of all transmitters and the electricity to customers are affected instantly.

vulnerable nodes, the dynamics of failure cascades unfold round by round as follows. The cascading process is deterministically in discrete rounds: in round t, all nodes that failed in round $t-1$ remain failed, and another node v fails if the total number of its failure neighbors is at least θ_u, i.e., $|N(u) \cap F_{t-1}| \geq \theta_u deg(u)$, in which F_{t-1} is the set of failure nodes before round $t-1$.

In addition, this model is also exhibited as one of the two main cascading models in the context of social science literature, which is referred to as Linear Threshold Propagation model. More importantly, this model belongs to the category of most contagion problems, such as models of failures in engineering systems, i.e., power grid [75], the Internet [5], or models of epidemics [53] and so on.

In the literature, some of the works assume that the thresholds θ_u are given as a part of the input. However, the thresholds are usually given as constant in communication networks due to the fixed load of each node. On the other hand, they are

generally not available and non-trivial to infer in social networks [39]. Therefore, instead of the random threshold, we use a simplified variation in which a node fails if a fraction θ of its neighbors failed in the previous round.

3.7.2 Cascading Random Failures

In order to analyze the pairwise connectivity in the residual power-law graph, we first provide the following theorem by extending from Theorem 3.6:

Theorem 3.6. *Consider the residual graph G' with expected degree sequence $\vec{d} = \{d_1, d_2, \dots, d_n\}$ ($\vec{y} = \{y_1, y_2, \dots, y_{\Delta'}\}$ represents the number of elements with value i in $\vec{d}$) and maximum degree Δ'. Given the following conditions w.r.t. the bounds of first-order degree summation*

$$d^1_{\min}(G') \leq \sum_{i=1}^{n} d_i = \overline{d}n \leq d^1_{\max}(G') \tag{3–4}$$

and the bounds of the second-order degree summation

$$d^2_{\min}(G') \leq \sum_{i=1}^{n} d_i^2 = \sum_{j=1}^{\Delta'} j^2 y_j \leq d^2_{\max}(G') \tag{3–5}$$

Then, we have the pairwise connectivity is a.s. at most $\frac{1}{2}n\left(c\Delta^{2\prime}\log n - 1\right)$. where $c = \frac{16}{2-\frac{d^2_{\max}(G')}{d^1_{\min}(G')}}$. Note that the bounds $d^1_{\min}(G')$ and $d^2_{\max}(G')$ are more important to assess the pairwise connectivity when the residual network is fragmented.

Proof. Here we only show the different parts from the proof of Theorem 3.6. After branching process method, we again focus on calculating $E(D_i)$, $E(B_i)$ and $E(\mathcal{L}_i)$ respectively. Note that we will just focus on the different steps from [78] in this proof. Let $\lambda = 2 - \frac{d^2_{\max}(G')}{d^1_{\min}(G')}$. We have,

$$E(D_i) \leq \frac{d^2_{\max}(G')}{d^1_{\min}(G')} = 2 - \lambda$$

Similar as in [78], we have $E(B_i) \geq 0$ due to $|N(u) \cap E_{i-1}| \geq 1$. Then

$$E(\mathcal{L}_i) \leq d_0 - \lambda(i-1)$$

is derive from the substitution of $E(D_i)$ and $E(B_i)$ into $\mathcal{L}_i - \mathcal{L}_{i-1} = D_i - B_i - 2$, where the initial node is assumed to have degree d_0.

Since the maximum degree in the residual graph G' is Δ', we have $|\mathcal{L}_j - \mathcal{L}_{j-1}| \leq \Delta'$. According to *Azuma's Martingale Inequality* [22],

$$\Pr\left[|\mathcal{L}_i - E(\mathcal{L}_i)| > \Omega\right] \leq 2e^{\frac{-\Omega^2}{2ie\Delta'^2}}$$

where $i = \frac{16}{\lambda^2}\Delta'^2 \log n = c\Delta'^2 \log n$ and $\Omega = \frac{\lambda}{2}i$. Since $E(\mathcal{L}_i) + \Omega \leq d_0 - \lambda(i-1) + \frac{\lambda}{2}i < 0$ for any d_0, we have

$$\Pr\left[\mathcal{T} > 2e^{\frac{-\Omega^2}{2i\Delta'^2}}\right] \leq \frac{2}{n^2}$$

Then, the probability that there is a non-failure node belonging to a connected component of size larger than $c\Delta'^2 \log n$ in graph G' is at most $o(1)$ and the upper bound pairwise connectivity in G' follows from Lemma 16 directly. □

In this subsection, we focus on investigating the expected degree sequence of the residual graph, along with its upper and lower bounds of first and second order degree summation.

Lemma 23. *When $p > \theta$, the upper bound $\max\{Pr_d^k\}$ of the probability Pr_d^k that a node v of degree k survives after $d > 0$ round cascades can be recursively computed using*

$$(1-p)\Big(\frac{1}{2}\exp\Big\{-2k\Big(1-\theta-\sum_{i=1}^{\Delta}\Phi_i \cdot \max\{Pr_{d-1}^i\}\Big)^2\Big\}\Big)$$

where $\Phi_i = \frac{1}{i^{\beta-1}\zeta(\beta-1)}$ is the probability that one of a neighbor for an arbitrary node has a node of degree i [78], and $Pr_0^i = 1-p$ for any degree i since each node randomly fails with the same probability p at the beginning.

Proof. Consider a node v of degree k and the probability Φ_i that v has a neighbor of degree i. We have

$$\Pr[v \text{ has } x_i \text{ neighbors of degree } i] = \frac{k!}{\prod_{i=1}^{\Delta} x_i!} \prod_{i=1}^{\Delta} {\Phi_i}^{x_i}$$

For a neighbor of v with degree i, it could either fail in round j with probability p_{ij} or survive after d rounds with probability $q_{i(d-1)}$ ($= \Pr^i_{d-1}$), that is, $\sum_{j=0}^{d-1} p_{ij} + q_{i(d-1)} = 1$. Let f_{ij} be the neighbors of degree i failed in round j and $s_{i(d-1)}$ be the neighbors of degree i survived after d round cascades. Note that the probabilities p_{ij} and $q_{i(d-1)}$ can be derived from the power-law random network model only based on the degree i of a node in a particular round j, along with the initial failure probability p of each node. Therefore, we have

$$\begin{aligned}
& \Pr[v \text{ survives at round } 0 \cap \\
& f_{ij} \text{ neighbors of degree } i \text{ fail in round } j \cap \\
& s_{i(d-1)} \text{ neighbors of degree } i \text{ survive after round } d-1] \\
= & \sum_{\substack{x_i \text{ neighbors of degree } i}} \Pr[v \text{ survives at round } 0 \cap \\
& f_{ij} \text{ neighbors out of } x_i \text{ of degree } i \text{ fail in round } j \cap \\
& s_{i(d-1)} \text{ neighbors out of } x_i \text{ of degree } i \text{ survive after} \\
& \text{round } d-1 \mid v \text{ has } x_i \text{ neighbors of degree } i] \\
& \cdot \Pr[v \text{ has } x_i \text{ neighbors of degree } i] \\
= & (1-p) \prod_{i=1}^{\Delta} \Big(\frac{x_i!}{\prod_{j=1}^{d-1} f_{ij}! s_{i(d-1)}} \prod_{j=0}^{d-1} p_{ij}^{f_{ij}} q_{i(d-1)}^{s_{i(d-1)}} \Big) \cdot \frac{k!}{\prod_i x_i!} \prod_{i=1}^{\Delta} {\Phi_i}^{x_i} \\
= & \frac{(1-p)k!}{\prod_i \prod_j f_{ij}! \prod_i s_{i(d-1)}} \prod_{i=1}^{\Delta} \prod_{j=0}^{d-1} (\Phi_i p_{ij})^{f_{ij}} \prod_{i=1}^{\Delta} (\Phi_i q_{i(d-1)})^{s_{i(d-1)}}
\end{aligned}$$

where the third step holds since the probability is equal to 0 when there exists some $x_i \neq \sum_{j=0}^{d-1} f_{ij} + s_{i(d-1)}$. Also, it is clear to see that $\sum_{i=1}^{\Delta} \sum_{j=0}^{d-1} \Phi_i p_{ij} + \sum_{i=1}^{\Delta} \Phi_i q_{i(d-1)} = 1$.

According to the cascading model, node v survives after d hop cascades if and only if less than θ fraction of its neighbors fail after $d-1$ round cascades. Therefore, we have

$$\begin{aligned}
\mathsf{Pr}_d^k &= \sum_{\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} f_{ij} \le \theta k} \frac{(1-p)k!}{\prod_i \prod_j f_{ij}! \prod_i s_{i(d-1)}} \\
&\prod_{i=1}^{\Delta}\prod_{j=0}^{d-1}(\Phi_i p_{ij})^{f_{ij}} \prod_{i=1}^{\Delta}(\Phi_i q_{i(d-1)})^{s_{i(d-1)}} \\
&= (1-p) \sum_{\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} f_{ij} \le \theta k} \binom{k}{\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} f_{ij}} \\
&(\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1}\Phi_i p_{ij})^{\sum_i \sum_j f_{ij}} (1 - \sum_{i=1}^{\Delta}\sum_{j=0}^{d-1}\Phi_i p_{ij})^{k - \sum_i \sum_j f_{ij}} \\
&\le (1-p)\big(\frac{1}{2}\exp\big\{-2k\big(\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1}\Phi_i p_{ij} - \theta\big)^2\big\}\big) \\
&= (1-p)\big(\frac{1}{2}\exp\big\{-2k\big(1 - \sum_{i=1}^{\Delta}\Phi_i q_{i(d-1)} - \theta\big)^2\big\}\big) \\
&\le \frac{1}{2}(1-p)\exp\big\{-2k\big(1-\theta-\sum_{i=1}^{\Delta}\Phi_i \cdot \max\{\mathsf{Pr}_{d-1}^i\}\big)^2\big\}
\end{aligned}$$

where the third step follows from the Hoeffding's inequality [76] and the last step follows from $\max\{\mathsf{Pr}_{d-1}^i\} < 1-p$ since the probability of the survival of a node has to be smaller than the probability it fails without cascading failures. □

Lemma 24. *When $p > \theta$, the lower bound $\min\{Pr_d^k\}$ of the probability Pr_d^k that a node v of degree k survives after $d > 0$ round cascades can be recursively computed using*

$$(1-p)\binom{k}{\theta k}\big(1 - \sum_{i=1}^{\Delta}\Phi_i \cdot \min\{Pr_{d-1}^i\}\big)^{\theta k}\big(\sum_{i=1}^{\Delta}\Phi_i \cdot \min\{Pr_{d-1}^i\}\big)^{(1-\theta)k}$$

where Φ_i is the probability that a node of degree k has a neighbor of degree i, and $Pr_0^i = 1-p$ for any degree i since each node randomly fails with the same probability p at the beginning.

Proof. According to the proof of Lemma 23, we know that

$$\begin{aligned}\Pr_d^k &= (1-p) \sum_{\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} f_{ij} \le \theta k} \binom{k}{\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} f_{ij}} \\ &\quad (\sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} \Phi_i p_{ij})^{\sum_i \sum_j f_{ij}} (1 - \sum_{i=1}^{\Delta}\sum_{j=0}^{d-1} \Phi_i p_{ij})^{k - \sum_i \sum_j f_{ij}} \\ &\ge (1-p)\binom{k}{\theta k} (1 - \sum_{i=1}^{\Delta} \Phi_i q_{i(d-1)})^{\theta k} (\sum_{i=1}^{\Delta} \Phi_i q_{i(d-1)})^{(1-\theta)k}\end{aligned}$$

Next, consider the function $f(x) = (1-x)^y x^{k-y}$ for some $0 \le x \le 1$ and $0 \le y \le k$. We have

$$\frac{df(x)}{dx} = (1-x)^{y-1} x^{k-y-1}(k - kx - y)$$

It is easy to see that $\frac{df(x)}{dx} > 0$ iff $k - kx - y > 0$. That is, $x < 1 - \theta$ when $y = \theta k$. Since $\min \Pr_{d-1}^i < q_{ij} < 1 - p$, we have $q_{ij} < 1 - \theta$ for any $0 \le j \le d-1$ when $p > \theta$. □

Lemma 25. *The expected number of node of degree $k' = k \sum_{i=1}^{\Delta} \Phi_i Pr_d^i$ in residual graph can be estimated as $\frac{e^\alpha}{k^\beta} Pr_d^k$ where the bounds of Pr_d^k is determined by Lemma 23 and 24.*

Proof. Consider a node of degree k in original graph G. After cascading failures, its degree can be estimated based on the survival of its neighbors. Particularly, for each neighbor, it has probability Φ_i to connect to a node of degree i. Moreover, a node of i in G will survive after d-round cascading failures with probability $\Pr_d^i$. Therefore, for a node of degree k in G, its degree in the residual graph can be estimated as $k \sum_{i=1}^{\Delta} \Phi_i \Pr_d^i$. On the other hand, each node of degree k in G has probability $\Pr_d^k$ to survive after cascades and there are $\frac{e^\alpha}{k^\beta}$ nodes of degree k in G. Therefore, the proof is complete. □

Using the lemma 25, we can obtain the following Theorem:

Theorem 3.7. *The expected first-order and second-order degree summation are*

$$\frac{e^\alpha}{\zeta(\beta-1)} \sum_{i=1}^{\Delta}\sum_{k=1}^{\Delta} \frac{1}{k^{\beta-1} i^{\beta-1}} Pr_d^i Pr_d^k$$

and

$$\frac{e^{\alpha}}{\zeta(\beta-1)^2}\sum_{i=1}^{\Delta}\sum_{j=1}^{\Delta}\sum_{k=1}^{\Delta}\frac{1}{k^{\beta-2}i^{\beta-1}j^{\beta-1}}Pr_d^i Pr_d^j Pr_d^k$$

where the bounds are determined by $\min\{Pr_d^k\}$ *and* $\max\{Pr_d^k\}$.

Proof. According to the definition of first-order degree summation, we have

$$\begin{aligned}
&\sum_{i=1}^{\Delta} k'\frac{e^{\alpha}}{k^{\beta}}\mathrm{Pr}_d^k \\
=\ &\sum_{i=1}^{\Delta}\left(k\sum_{i=1}^{\Delta}\Phi_i\mathrm{Pr}_d^i\right)\frac{e^{\alpha}}{k^{\beta}}\mathrm{Pr}_d^k \\
=\ &e^{\alpha}\sum_{i=1}^{\Delta}\sum_{i=1}^{\Delta}\frac{1}{i^{\beta-1}\zeta(\beta-1)}\frac{1}{k^{\beta-1}}\mathrm{Pr}_d^i\mathrm{Pr}_d^k \\
=\ &\frac{e^{\alpha}}{\zeta(\beta-1)}\sum_{i=1}^{\Delta}\sum_{k=1}^{\Delta}\frac{1}{k^{\beta-1}i^{\beta-1}}\mathrm{Pr}_d^i\mathrm{Pr}_d^k
\end{aligned}$$

Again, according to the definition of second-order degree summation, we have

$$\begin{aligned}
&\sum_{i=1}^{\Delta} k'\frac{e^{\alpha}}{k^{\beta}}\mathrm{Pr}_d^k \\
=\ &\sum_{i=1}^{\Delta}\left(k\sum_{i=1}^{\Delta}\Phi_i\mathrm{Pr}_d^i\right)^2\frac{e^{\alpha}}{k^{\beta}}\mathrm{Pr}_d^k \\
=\ &\frac{e^{\alpha}}{\zeta(\beta-1)^2}\sum_{i=1}^{\Delta}\sum_{j=1}^{\Delta}\sum_{k=1}^{\Delta}\frac{1}{k^{\beta-2}i^{\beta-1}j^{\beta-1}}\mathrm{Pr}_d^i\mathrm{Pr}_d^j\mathrm{Pr}_d^k
\end{aligned}$$

□

3.7.3 Numerical Analysis

Here we show that our theoretical analysis consists well with the simulation result. Particularly, we generate power-law networks using igraph package [26] and test on the synthetic networks with different parameters, exponential factor β and network size n. Due to the similar results using distinct parameters, we only provide the result with $\beta = 1.5$ and $n = 250$ as in Fig. 3-7. As revealed in Fig. 3-7, apart from the surprising agreement between our analysis and simulation, we also find that power-law networks

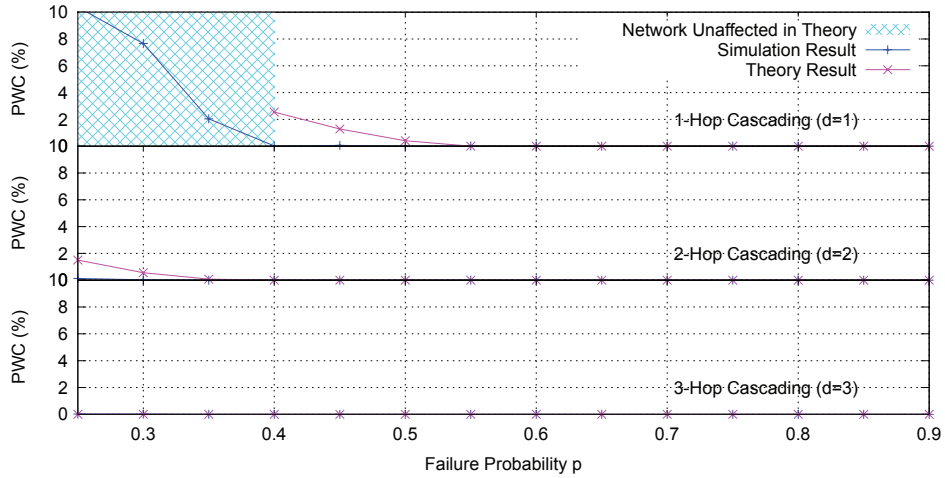


Figure 3-7. Numerical Analysis in Power-Law Networks ($\beta = 1.5$, $n = 250$). We plot the three cascading hops and find that our analysis (pink plots) approximates the simulation of the total pairwise connectivity (PWC) after cascading failures surprisingly well, in both cases that power-law networks are a.s. unaffected (PWC$\propto n^2$) and a.s. fragmented.

are no longer robust under random failures when cascading failures occur. For example, when each node is attacked with probability only 0.4, the network is a.s. fragmented only after 1-hop propagation. This transition happens only when the probability equal to 0.2 if failures can cascade 2 hops and almost vanishes when allowing more hop cascades.

3.8 Related Works

There are a great number of studies regarding the tolerance of real-world networks against failures and attacks using different metrics. Edge vulnerability in metabolic networks was studied by Kaiser *et al.* with respect to the average shortest path and the clustering coefficient [50]. For the sake of power grid networks, Albert *et al.* [3] investigated their vulnerability by measuring the loss of connectivity under various threats, including random, cascading, load-based and degree-based nodal failures. The disruption of vital interstate systems was assessed by Matisziw *et al.* [63] according to the available number of compromised $s - t$ flows. Cohen *et al.* [23] showed the resilience of Internet to the random breakdown of the nodes based on percolation theory. In [72], Satorras *et al.* also revealed that the random uniform immunization of

individuals cannot lead to the eradication of communications in complex social networks using the reduced prevalence rate. Doyle *et al.* [30] and Sydney *et al.* [81], using a novel metric *ELASTICITY*, explored that Internet topologies are less affected by both random and targeted attacks than the power-law networks. In general, the robustness of other complex networks was studied in [46, 47] using algebraic connectivity, i.e., the second-smallest eigenvalue of the Laplacian matrix of a graph. Recently, from a different perspective, Alderson *et al.* [7] focused on the role of organization and design in terms of the complexity in highly organized technological and biological systems.

More generally, Albert *et al.* [5] first compared the robustness of complex systems with the power-law and exponential properties. By measuring the diameters, the relative size of the largest cluster and the average size of the isolated clusters, the power-law networks are empirically observed to tolerate failures to a surprising degree but their survivability decreases rapidly under attacks after comparing them with exponential networks. Later on, Holme *et al.* [43] further investigated the degree of harms to power-law networks under different strategies of attacks. Unfortunately, all these observations are derived from experiments and lack their theoretical foundations.

CHAPTER 4
OPTIMIZATION OF POWER-LAW NETWORKS

In this chapter, we investigate the tradeoff impact of maintenance costs and robustness guarantee on the power-law networks. In particular, we focus on the power-law networks with $\beta < 2.9$, which have been discovered to tolerate random failures to an extreme high degree. In addition, since Fig. 3-3, 3-4 and 3-5 already revealed that power-law networks can tolerate preferential attacks if they can tolerate degree-centrality attacks when $\beta < 2.9$, we focus on the guarantee of their functionality under degree-centrality attacks. We study the practical communication networks and social networks respectively to explore the underlying reasons of their real-world network topologies.

On the other hand, we show the NP-hardness to detect these critical elements in power-law networks, along with two algorithms in two different cases: element failures and cascading failures. The effectiveness of our algorithms are evaluated on both synthetic power-law networks and real-world networks.

4.1 Design Optimization of Power-law Networks

The above vulnerability assessments give us a belief that power-law networks are more robust when β is smaller. However, a majority of real-world networks usually have their exponential factor β ranging from 2 to 2.5 rather than some small β approaching 1 or even less. The questions are intuitively raised: Is it better if real-world networks are denser such that they can be more robust? What causes them to be sparser than our expectation? Does there exist some potential optimization factors?

To address these questions, in this section, we investigate the tradeoff impact of maintenance costs and robustness guarantee on the power-law networks. In particular, we focus on the power-law networks with $\beta < 2.9$, which have been discovered to tolerate random failures to an extreme high degree. In addition, since Fig. 3-3, 3-4 and 3-5 already revealed that power-law networks can tolerate preferential attacks if they

can tolerate degree-centrality attacks when $\beta < 2.9$, we focus on the guarantee of their functionality under degree-centrality attacks. We study the practical communication networks and social networks respectively to explore the underlying reasons of their real-world network topologies.

4.1.1 Communication Networks

In the design of communication networks, such as the Internet, telecommunication networks and so on, we are required not only to guarantee their functionality but reduce the maintenance costs as well. Among various network performance metric, i.e., delay, packet loss, throughput, etc., the guarantee of its connectivity is of the high priority. That is, a real-world network only need sufficient number of links to guarantee its functionality, and its other performance metric can be guaranteed by adjusting its capacity planning [59]. In particular, we consider the costs including the link costs and the protection costs for critical nodes. Since the nodes with degree and betweenness centrality are closely correlated in non-fractal power-law networks [56], here we consider the critical nodes to be degree-centrality nodes.

To formulate the optimization function for power-law networks in communication networks, we first prove the following Lemma 26 by considering the worst case with respect to the robustness of power-law networks. That is, as mentioned above, after protecting the degree-centrality nodes, power-law networks a.s. remains highly-connected (its total pairwise connectivity is a.s. $\Theta(n^2)$) even though all other nodes are failed.

Lemma 26. *Let G_{cp} be the residual graph of $G_{(\alpha,\beta)}$ only consisting of the protected degree-centrality nodes (the nodes of degree larger than x_0), we have*

- *The pairwise connectivity $\mathbb{P}$ is a.s. $\Theta(n^2)$ if $x_0 < \max\left\{x_0 \middle| \frac{1}{\zeta(\beta-1)} \frac{\left(\sum_{x=x_0+1}^{\Delta} \frac{1}{x^{\beta-1}}\right)^2}{\sum_{x=x_0+1}^{\Delta} \frac{1}{x^{\beta}}} > 1\right\}$;*
- *The pairwise connectivity $\mathbb{P}$ is a.s. at most $\frac{1}{4}n^{\frac{3}{2}} \log n$ if $x_0 > \min\left\{x_0 \middle| \frac{1}{\zeta(\beta-1)} \sum_{x=x_0+1}^{\Delta} \frac{1}{x^{\beta-2}} < 1\right\}$.*

Proof. Consider that we protect only all nodes of degree larger than x_0 and all other nodes are failed. Similar as in Corollary 10, the expected degree sequence can be written as

$$\mathsf{E}(y_i^{cp}) \doteq \frac{e^\alpha}{i^\beta}\left(\frac{1}{\zeta(\beta-1)}\sum_{x=x_0+1}^{\Delta}\frac{1}{x^{\beta-1}}\right)^\beta$$

The rest of proof is the same as Theorem 3.4. □

In order to guarantee the functionality of a power-law network, we take the above Lemma 26 as the condition. In the meanwhile, we aim to minimize the maintenance costs, which include the link costs and critical node protection costs. In detail, we consider the following cost functions:

- *Link Costs:* Consider a link (u, v) in $G_{(\alpha,\beta)}$, its link cost is heavily dependent on the number of messages it transmits according to [62]. Another crucial factor for the link cost is its capacity flow [79]. Since the nodes with degree and betweenness centrality are closely correlated in non-fractal power-law networks [56], we consider the link cost proportional to the average of the degrees of its two endpoints.
- *Critical Node Protection Costs:* In terms of the critical nodes in $G_{(\alpha,\beta)}$, apart from their degrees, their protection costs are also closely related with the network density. According to [79], the costs will rise with the increase of density since it enlarges the demand of message exchanges. In addition, as investigated in [62], the chain reaction leads to the roughly exponential increase of costs, we consider the cost $\gamma(x)$ to protect a node of degree x as $a * x^{b/\beta}$ for some constant a and b.

Therefore, we can confidently formulate the following *Mixed Linear Programming* (MIP), with two variables x_0 and β, as

$$\begin{aligned}
\min\quad & \tfrac{1}{2}\textstyle\sum_{x=1}^{\Delta}\frac{e^\alpha}{x^\beta}x + \sum_{x=x_0+1}^{\Delta}\frac{e^\alpha}{x^\beta}\gamma(x)\\
\text{s.t.}\quad & \frac{1}{\zeta(\beta-1)}\frac{\left(\sum_{x=x_0+1}^{\Delta}\frac{1}{x^{\beta-1}}\right)^2}{\sum_{x=x_0+1}^{\Delta}\frac{1}{x^\beta}} > 1\\
& x_0 \in Z^+, x_0 \le \Delta\\
& \beta > 0
\end{aligned} \tag{4–1}$$

Note that we omit the proportional constant of link cost since it does not affect the optimization of total maintenance costs.

4.1.2 Social Networks

As we mentioned at the beginning of this paper, one of the main threats in social networks is the malware propagations [87]. Thus, apart from the factors in [11], the containments of these malicious spreading become another crucial factor of the sparsification of social networks. In other words, when an individual is infected, we want to minimize the expected number of total infected users, which can be realized by immunizing critical users beforehand. Therefore, the minimization of immunization costs becomes an urgent need.

Thus, in order to formulate the optimization function for power-law networks in social networks, we first investigate the upper bound of expected size of a connected component after protecting the critical users, which are again referred to as the degree-centrality nodes. That is, we focus on the size of connected components on residual network after removing such immunized users. By defining the residual graph G_s to be the residual power-law graph $G[V \setminus S]$ after immunizing individuals in S, the following Theorem 4.1 gives the bound of expected size of a connected component on G_s.

Theorem 4.1. *In the residual graph G_s of $G_{(\alpha,\beta)}$, the expected size of a connected component $\bar{c}$ is a.s. upper bounded by $O\left(n^{\frac{1}{4}}\right)$ when $\tilde{d}_s < 1$, that is, $x_0 < \max\left\{x_0 \middle| \frac{1}{\zeta(\beta-1)} \sum_{x=1}^{x_0} \frac{1}{x^{\beta-2}} < 1\right\}$.*

Proof. Consider the connected components $c_1, c_2, \ldots, c_k$ in G_s, their expected size $\bar{c}$ can be written as $\frac{1}{k} \sum_{1 \le i \le k} c_i$. According to [21], all connected components a.s. have volume at most $C'\sqrt{n}$ for some constant C' when $\tilde{d} < 1$. Therefore, the number of connected components is at least $C\sqrt{n}$ where $C = 1/C'$. Supposing that $\bar{c} \ge \gamma n^{\frac{1}{4}}$ for some constant γ with probability ρ, the probability that any random pair of nodes are in the same component can be lower bounded by

$$\frac{1}{n^2 \overline{d}_s^2} \rho \sum_{1 \le i \le k} c_i^2 \ge \frac{1}{n^2 \overline{d}_s^2} \rho \bar{c}^2 k \ge \frac{1}{n^2 \overline{d}_s^2} \rho C \gamma^2 n$$

On the other hand, according to F. Chung *et al.* [21], we know that the probability for any random pair of nodes belonging to the same component is upper bounded by

$$\frac{\tilde{d}_s^2}{(1-\tilde{d}_s)n\overline{d}_s}$$

Combining the above two bounds, we know

$$\frac{1}{n^2\overline{d}_s^2}\rho C\gamma^2 n \leq \frac{\tilde{d}_s^2}{(1-\tilde{d}_s)n\overline{d}_s}$$

which implies that

$$\rho \leq \frac{\overline{d}_s\tilde{d}_s^2}{C\gamma^2(1-\tilde{d}_s)}$$

That is, by choosing C to be $\log n$, with probability at least $1-o(1)$, the expected size $\bar{c}$ of connected components is a.s. at most $O\left(n^{\frac{1}{4}}\right)$. □

Again, consider the above lemma as the condition and the same protection cost function of critical users $\gamma(x) = a * x^{b/\beta}$, we formulate the following mixed linear programming, with two variables x_0 and β, in order to make sure that the expected size of connected components in the residual power-law networks is no larger than $O\left(n^{1/4}\right)$.

$$\begin{aligned} \min \quad & \textstyle\sum_{x=x_0+1}^{\Delta} \frac{e^\alpha}{x^\beta}\gamma(x) \\ \text{s.t.} \quad & \textstyle\frac{1}{\zeta(\beta-1)}\sum_{x=1}^{x_0}\frac{1}{x^{\beta-2}} < 1 \\ & x_0 \in Z^+, x_0 \leq \Delta \\ & \beta > 0 \end{aligned} \tag{4–2}$$

4.1.3 Optimal Range of Exponential Factor β

For the sake of communication networks, consider the practical range of protection costs from 0 to x^9/β for a node of degree x (that is $b \in [0,3]$), Fig. 4-1 reveals the relation between maintenance costs and optimal β according to MIP (4–1). As one can see, the optimal β is from 1.8 to 2.5 no matter how large the constant b is, the exponential factor β is no less than 1.8. (Note that the curve is invariant for distinct

network sizes since the effect of light-tailed elements in riemann zeta function can be neglected.)

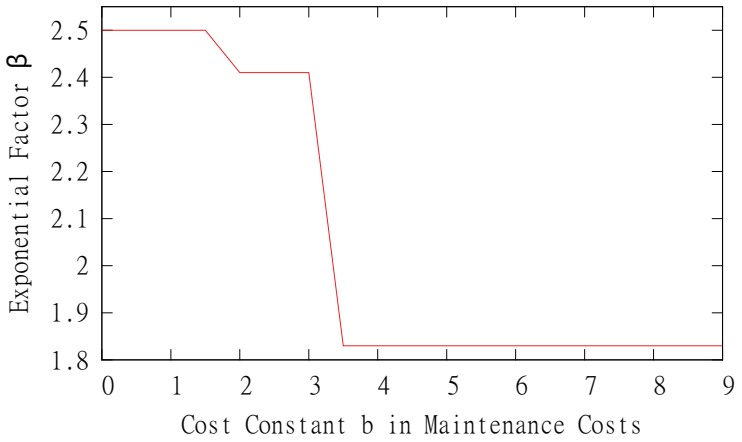


Figure 4-1. Optimal Robust Communication Networks

Fig. 4-2 reports that the optimal range of β is from 2.3 to 2.4 in social networks according to MIP (4–2). We observe that the increase of b does not really affect the range of β and the curve also remains invariant with respect to different network sizes.

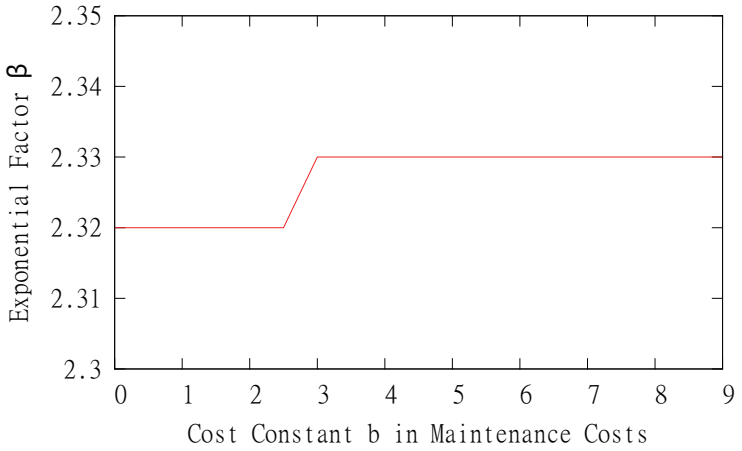


Figure 4-2. Optimal Robust Social Networks

In summary, the analysis on both communication networks and social networks give us a reasonable explanation of the topology in real-world power-law networks, that is, the best range of the exponential factor β is $[1.8, 2.5]$. In other cases, the network

maintenance cost either becomes very expensive when $\beta < 1.8$, or the network robustness is unpredictable when $\beta > 2.5$ due to its dependence on the specific attacking strategy.

4.2 Critical Elements Detection in Power-law Networks

In this section, we study two practical optimization problems namely *Critical Link and Node Disruptor* (CLD and CND), to assess the network vulnerability when a given number of network elements (links or nodes) fail undesirably. We refer to these elements as *critical links and nodes* hereafter.

Definition 23 (Critical Link Disruptor)**.** *Given an integer k and a weighted undirected graph $G = (V, E, W)$, the problem asks for a weight-bounded subset of critical links $S \subset E$, i.e. $\sum_{(i,j)\in S} w_{ij} \leq k$, whose removal minimizes the total pairwise connectivity in $G[E \setminus S]$.*

Definition 24 (Critical Node Disruptor)**.** *Given an integer k and a weighted undirected graph $G = (V, E, W)$, the problem asks for a weight-bounded subset of critical nodes $S \subset V$, i.e. $\sum_{v_i\in S} w^i \leq k$, whose removal minimizes the total pairwise connectivity in $G[V \setminus S]$.*

Moreover, taking into account the cascading failures, we define another problem focusing on the detection of critical nodes, called *Cascading Vulnerability Node Detection* (CVND) problem, as follows:

Definition 25 (Cascading Critical Node Disruptor)**.** *Given two integers k, d, a fractional number $\theta \in (0, 1)$ and an undirected graph $G = (V, E)$. Let $P(S)$ be total pairwise connectivity of residual graph G after the d-hop cascading failures caused by the initial removal of the set of nodes $S \in V$. The CVND problem asks for k most vulnerable nodes such that $P(S)$ is minimized.*

4.2.1 Hardness of Detecting Critical Links and Nodes

In this subsection, we show that CLD and CND problems are NP-hard, which denies the existence of a prompt optimal solution.

Lemma 27 (Ferrante *et al.* [35]). *Let $G_1 = (V_1, E_1)$ be a simple graph with n nodes and $\beta \geq 1$. For $\alpha \geq \max\{4\beta, \beta \log n + \log(n+1)\}$, we can construct a power-law graph $G = G_1 \cup G_2$ with exponential factor β and the number of nodes $e^{\alpha}\zeta(\beta)$ by constructing a bipartite G_2 as a maximal component in G.*

Lemma 28. *The clique separator (CS) problem (which is defined as given an undirected graph $G = (V, E)$, find a minimum set of links $S \subseteq E$ such that the connected components of $G[E \setminus S]$ are cliques, each has size at least 3) is* NP*-hard.*

Theorem 4.2. *The CLD problem is* NP*-hard on power-law graphs even if all nodes have unit weights.*

Proof. Consider the decision version of CLD that asks whether an undirected graph $G = (V, E)$ contains a set of links $S \subset E$ of size k such that the pairwise connectivity in residual graph $G[E \setminus S]$ is at most c for a given positive integer c. To prove that CLD on power-law graphs is in *NP*-hard, we reduce the clique separator (CS) to it. After constructing a power-law graph $G' = G \cup G_b$ where the bipartite graph $G_b = (U_b, V_b; E_b)$ is a maximal component in G' according to Lemma 27, we show that there is a CS of size k in G iff G' has a CLD S' of size k' such that the pairwise connectivity of $G'[E' \setminus S']$ is at most c, where $k' = k + |E_b| - |M_b|$ and $c = |E| - k + |M_b|$. Note that M_b is the links in the maximum matching of G_b.

First, suppose $S \subseteq V$ is a clique separator of G with $|S| = k$. We have the pairwise connectivity in $G[E \setminus S]$ to be $|E| - k$ since all components in this graph are cliques. Since the maximum matching on G_b can be found in polynomial time using Hopcroft-Karp algorithm [44], the pairwise connectivity on G' is c after removing additional $|E_b| - |M_b|$.

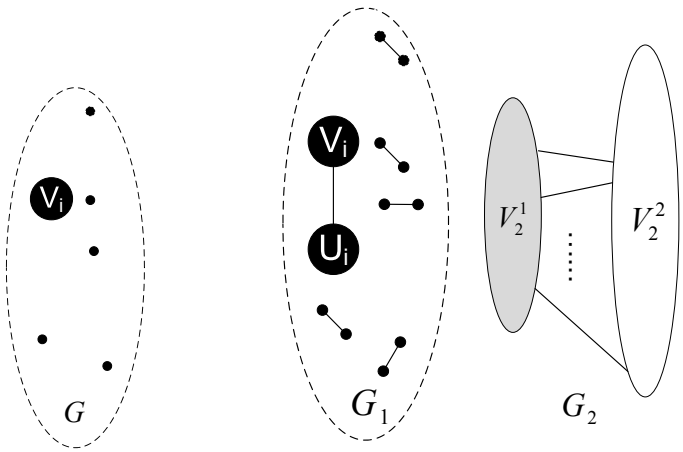
Conversely, suppose that $S' \subset V'$ is a CLD of G' with size k'. Note that $S' = A \cup S_b$, where A and S_b are CLD on G and G_b respectively. We show that the number of critical links S_b in G_b is $|E_b| - |M_b|$. If $|S_b| < |E_b| - |M_b|$, the pairwise connectivity of G_b increases by at least two when adding one more link onto the maximum matching. On the other hand, the removal of l links on G can reduce the pairwise connectivity at most l after

removing the CS k. If $|S_b| > |E_b| - |M_b|$, the pairwise connectivity of G_b reduce by one when removing one more link from the maximum matching. Meanwhile, a link added onto the residual graph of G will increase the pairwise connectivity at least one if it connects two independent nodes and at least 3 if it has one endpoint belonging to some component in the residual graph of G. Thus, we have $S_b = |E_b| - |M_b|$ and it is easy to verify that A is a CS of G. □

Theorem 4.3. *The CND problem is* NP*-hard on power-law graphs even if all nodes have unit weights.*

Proof. Consider the decision of CND that asks whether a graph $G = (V, E)$ contains a set of nodes $S \subset V$ of size k such that the pairwise connectivity in $G[V \setminus S]$ is at most c for a given positive integer c. To prove that CND on power-law graphs is in *NP*-hard, we reduce the vertex cover (VC) to it. Let an undirected graph $G = (V, E)$ where $|V| = n$ and a positive integer $k \leq n$ be any instance of VC. We construct a power-law graph $G' = (V', E')$ as follows. First, for each node $v_i \in V$ on graph G, we add one additional node u_i onto it, which we call G_1 and $V_1 = V \cup U$ where $U = \{u_i\}$. Then according to Lemma 27, a power-law graph $G' = (V', E')$ can be constructed by embedding G_1 and a bipartite graph $G_2 = (V_2^1, V_2^2; E_2)$ where V_2^1, V_2^2 are two sets of disjoint nodes in G_2 and $\alpha \geq \max\{4\beta, \beta \log(2n) + \log(2n + 1)\}$ with some specific β. Note that V_2^1 and V_2^2 are marked gray and white separately as shown in Fig. 4-3. We show that there is a VC of size k in G iff G' has a CND S' of size k' such that the pairwise connectivity of $G'[V' \setminus S']$ is at most c, where $k' = k + \min\{|V_2^1|, |V_2^2|\}$ and $c = n - k$.

First, suppose $S \in V$ is a vertex cover of G with $|S| = k$. Therefore, G has a vertex cover S of size k iff $G \cup G_2$ has a vertex cover S' of size $k + \min\{|V_2^1|, |V_2^2|\}$ since VC is polynomially solvable in any bipartite graphs according to Kõnig's Theorem [49]. Then, after removing S' from G', we only have all disjoint links (v_i, u_i) left where $v_i \notin S$. Therefore, the pairwise connectivity on power-law graph G' is $n - k$, which is equal to c.



A An instance B Reduced graph $G' = G_1 \cup G_2$

Figure 4-3. An example of CND reduction on PLGs. For simplicity, we just draw the nodes in G and its newly added nodes and links.

Conversely, suppose that $S' \subset V'$ with $|S'| = k'$ is a CND of G', that is, the total pairwise connectivity of $G'[V' \setminus S']$ is at most c. First, if $u_i \in S'$, it is easy to verify that replacing u_i with any v_i will further decrease the pairwise connectivity. Since $|S'| = k' = k + \min\{|V_2^1|, |V_2^2|\}$, we can easily modify S' to be a vertex cover of $G \cup G_2$, where the total pairwise connectivity on G' is at most $c = n - k$. Thus $S' \cap V$ is a VC of G. □

4.2.2 HILPR Approach

Apart from the above theoretical hardness results for CLD and CND, these two problems are usually even harder to be approximated. The pairwise connectivity can either remain $O(n^2)$ for CLD in dense networks even when k is large, or reach 0 for CND when k is larger than the size of vertex cover. In this section, we present our solution, a *Hybrid Iterative Linear Programming Rounding* (HILPR) algorithm to both CLD and CND problems. In a big picture, HILPR formulates CLD and CND under Integer Linear Programming (ILP) formulations, and then solves them using an iterative rounding technique. In addition, HILPR also takes into account the local search and

constraint pruning techniques in order to further improve its efficiency and reduce its time complexity.

4.2.2.1 Integer linear programming formulation

Critical Link Disruptor

For each pair of nodes $i, j \in V$, we define an indicator variable u_{ij} as:

$$u_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}$$

Then we have the following ILP:

$$\begin{aligned} \min \quad & \sum_{i,j \in V} u_{ij} \\ \text{s.t.} \quad & u_{ij} + u_{jh} - u_{hi} \leq 1 \qquad \forall i, j, h \in V \\ & \sum_{(i,j) \in E} w_{ij}(1 - u_{ij}) \leq k \\ & u_{ij} \in \{0, 1\} \end{aligned} \tag{4–3}$$

where the objective is to minimize the total pairwise connectivity. The first constraint imposes the triangular connectivity. That is, if node i and j are connected, node j and h are connected, node i and h have to be connected. The second constraint means that the total weight of all deleted links has to be at most k. We note that for an edge $(i, j) \in E$, if $u_{ij} = 0$ in the ILP solution, then that link (i, j) is a critical link.

Critical Node Disruptor

For CND, we simply extend the above IP formulation for CLD in (4–3) as

$$\begin{aligned} \min \quad & \sum_{i,j \in V} u_{ij} \\ \text{s.t.} \quad & v_i + v_j + u_{ij} \geq 1 \qquad \forall (i, j) \in E \\ & u_{ij} + u_{jh} - u_{hi} \leq 1 \qquad \forall i, j, h \in V \\ & \sum_{i \in V} w^i v_i \leq k \\ & v_i \in \{0, 1\}, u_{ij} \in \{0, 1\} \end{aligned} \tag{4–4}$$

where v_i is further defined as

$$v_i = \begin{cases} 1, & \text{if node } i \text{ is deleted (i.e., critical nodes)} \\ 0, & \text{otherwise} \end{cases}$$

The first additional constraint guarantees that at least one endpoint of a link has to be deleted if its two endpoints are disconnected in the optimal solution. Other constraints are carried out as CLD. We further constrain k in CND to satisfy Lemma 29 for unweighted graphs to avoid the zero pairwise connectivity, that is, all nodes in network are independent.

Lemma 29. *For an unweighted graph G, the optimal pairwise connectivity of CND is larger than 0 if $k < |E|/\Delta$, where Δ denotes the maximum degree in G.*

Proof. We prove this using the contradiction method. Assume the optimal pairwise connectivity is 0, that is $\sum_{i,j \in V} u_{ij} = 0$, we have $u_{ij} = 0$ for any single link (i, j). Therefore, $v_i + v_j \geq 1$ according to the first constraint in LP (4–5). Hence we have $\sum_{i \in V} d_i v_i \geq |E|$ by adding this up for all links. Note that we assumed $k < |E|/\Delta$, so $\sum_{i \in V} d_i v_i \leq \Delta \sum_{i \in V} v_i \leq k\Delta < |E|$, which draws a contradiction. □

4.2.2.2 Hybrid iterative lp rounding algorithm

The basic idea of our HILPR algorithm consists of three main steps: (1) Relaxing the integral constraints of the above ILP to obtain the corresponding LP; (2) Iteratively solving the LP by replacing k in it with some experimental parameter $\gamma < k$ and rounding the corresponding fractional solutions of which have weights at most γ to integers; and (3) Performing the local search to further optimize the solutions. The detailed description is shown in Algorithm 7.

Specifically, in each iteration, we solve the LP after setting $k = \gamma$. Let u_{ij}^* and v_i^* be the optimal (fractional) solutions after solving the LP for the CLD and CND problems respectively. In the CLD problem, we round X smallest variables u_{ij}^* to 0 such that $\sum_{u_{ij} \in X} w_{ij} \leq \gamma$. Likewise, we round the Y largest variables v_i^* into 1 for the CND problem

such that $\sum_{v_i \in Y} w^i \leq \gamma$. In the next iteration, the graph will first be updated according to the previous rounding results, i.e., the identified critical links or nodes will be removed from the graph. Then LP will be reformulated according the new residual graph. The algorithm terminates when the total weight of all identified critical links (or nodes) reaches to k.

Algorithm 7: HILPR for a given γ

Input : Graph $G = (V, E)$, an integer k, γ
Output: The set of critical links/nodes S

$S \leftarrow \emptyset$;
// Iterative LP Rounding
while $k > 0$ **do**
  if $k < \gamma$ **then**
    $\gamma = k$;
  end
  else
    Use Constraint Pruning to solve the LP formulation with γ;
    $k \leftarrow k - \gamma$;
  end
  $S' \leftarrow X$ links with smallest u_{ij}^* such that $\sum_{u_{ij} \in X} w_{ij} \leq \gamma$ (in case of CND, $S' \leftarrow Y$ nodes with largest v_i^* such that $\sum_{v_i \in Y} w^i \leq \gamma$);
  $S \leftarrow S \cup S'$;
  $G \leftarrow G[E \setminus S']$;
end
// Local Search
$S^* \leftarrow S$;
foreach *element* $e \in S$ **do**
  `Swapping(`e`)` (Algorithm 8);
end
$S \leftarrow S^*$;
return S;

In the end, the HILPR algorithm further deploys a meta-heuristic approach [38] to enhance the solution S obtained by the iterative rounding step mentioned above. The detail of this local search is shown in the last part of Algorithm 7. Take a CLD as an example. For each link e in the solution S, we do the local swapping between e and each $e' \in N(e)$ to obtain the new solution $S' = S \cup \{e'\} \setminus \{e\}$. Let $f(G, S)$ be the pairwise

connectivity function of $G[E \setminus S]$. If $f(G, S') < f(G, S)$, we replace e by e' in the solution, set S to be S' and recursively do the local search on e'. The recursive procedure stops until no more improvement can be achieved by the local search, i.e., $f(G, S') \geq f(G, S)$. The whole algorithm stops until all links in S are visited. Similarly, the local search on CND can be achieved by recursively checking the neighbor nodes.

Algorithm 8: Swapping(e)

```
S* ← S* \ {e};
if ∃e' ∈ N(e) such that f(G, S') < f(G, S) where S' ← S \ {e} ∪ {e'} then
    S* ← S';
    Swapping(e');
end
```

We note that the LP formula of CLD and CND each has $O(n^3)$ constraints owing to the triangle inequality constraints. To improve the running time of our algorithm during solving the LP, we further propose the constraint pruning technique to eliminate the inactive constraints according to the following lemma. As a result, the number of active constraints on triangle inequality in equation (4–3) and (4–5) can be reduced to $O(n^2)$ according to the constraint pruning technique.

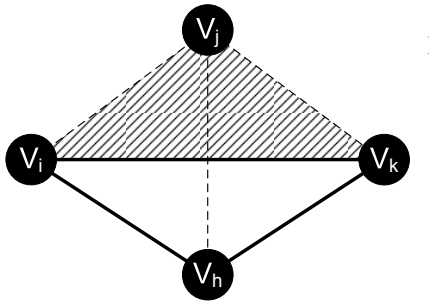


Figure 4-4. Triangle inequality constraints

Consider a four-tuple triangle inequality (i, j, k), (i, j, h), (i, k, h) and (j, k, h), all constraints are satisfied at the beginning. That is, as shown in Fig. 4-4, $u_{ij} + u_{jh} - u_{hi} \leq 1$ and $u_{jk} + u_{hk} - u_{jh} \leq 1$ for the tuple (i, j, h) and (j, k, h) respectively. In the case that the triangle inequality of the tuple (i, j, k) is tight, shown as shadow in Fig. 4-4, that is,

$u_{ij} + u_{jk} - u_{ik} = 1$, we have

$$u_{hi} \geq u_{ij} + u_{jh} - 1 \geq u_{ij} + u_{jk} + u_{kh} - 2 = u_{ik} + u_{kh} - 1$$

Thus, the triangle inequality of the tuple (i, k, h) is satisfied, shown as bold in Fig. 4-4. Once the triangle inequality of the tuple (i, j, k) is tight, the triangle inequality of the tuple (i, k, h) will be satisfied for all nodes h. Since the number of triangle inequality constraints is $3\binom{n}{3} = O(n^3)$, the number of active constraints will be $O(n^3)/n = O(n^2)$ after pruning process.

4.2.2.3 Performance evaluation

Performance of the HILRP Algorithm

The three networks we use to evaluate the performance of our proposed HILPR algorithm are described as follows:

1. The real terrorist network compiled by Krebs [57] with 62 nodes and 153 links, which reflects the relationship between the terrorists involved in the terrorism attacks of Sep. 11, 2001. This experiment attempts to evaluate the performance of HILPR on a real-world social network. In order to breakdown the terrorist network, we can capture the individuals corresponding to the critical nodes identified by HILPR.

2. Waxman network topology, a widely-accepted Internet AS topological model, is generated by the well-known BRITE [64].

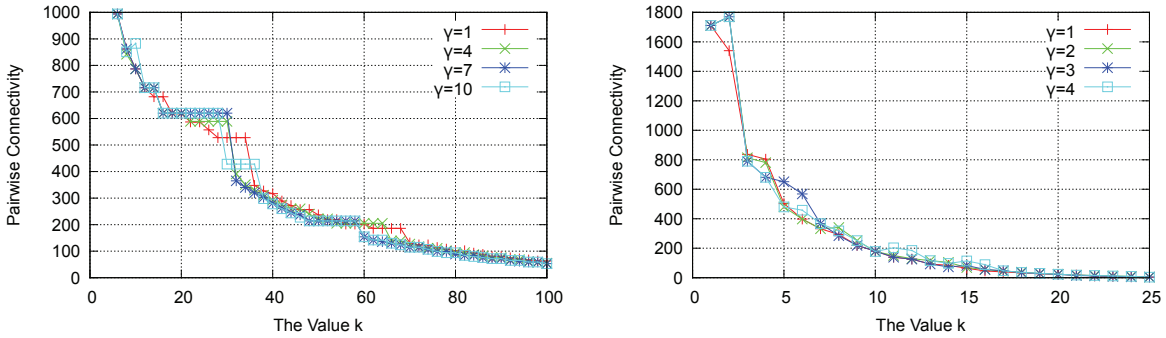
3. Power-law network topology, generated by Barabási graph generator [1], has been discovered as one of the most remarkable properties in many large-scale networks such as the Internet and the social networks.

To keep the similar density as the real terrorist network and also show the comparison with optimal solutions, we use the instance with 70 nodes and 140 links. We generate 100 instances for both Waxman and power-law models and show the average results.

In order to show the effectiveness of our proposed HILPR algorithm, we compare it with the optimal solution obtained by solving the ILP directly. We also compare HILPR with two centrality approaches: degree centrality (DC) and betweenness centrality (BC), which are often used in network analysis [17]. In DC, the k links and nodes of largest

degrees are selected as critical links and nodes, where the degree of a link (u, v) is defined as $d(u) + d(v)$. Similarly, in BC, the k links and nodes with largest betweenness are selected as critical links and nodes, where the betweenness of a link or a node is defined as the number of shortest paths among all pairs of nodes that passes through it. For CND, we further compare HILPR with CNLS approach proposed by Arulselvan *et al.* [9], which also aims to minimize the pairwise connectivity.

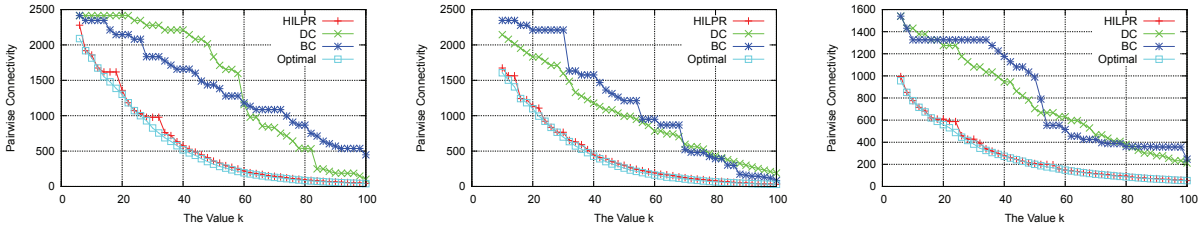
As the only free parameter in our algorithm, we first compare the impacts of different γ values in our experiments such that we can balance the solution quality and running time by carefully selecting this experimental value γ. As illustrated in Fig. 4-5, the results returned by our algorithm are very close solutions. Thus, we use $\gamma = 1$ for CND due to its slightly better performance, and $\gamma = 5$ for CLD to reduce the running time since the number of critical links is usually larger compared with critical nodes. Next, we show that our HILPR approach returns a very-near optimal solution and outperforms other approaches.



A Critical Links B Critical Nodes

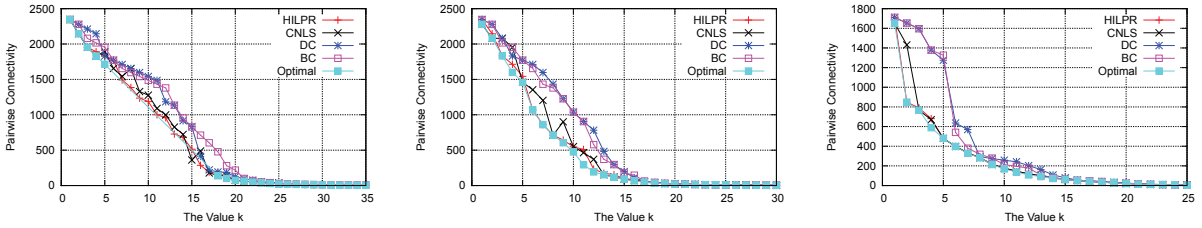
Figure 4-5. The performance of HILPR using different γ in terrorist network

Fig. 4-6 and Fig. 4-7 report the comparison of the above HILPR algorithm and centrality algorithms for CLD and CND on the above three different networks. In these figures, we notice that the solution of HILPR algorithm is very closely approaching the optimal solution for both CLD and CND on all these three networks (Note that a portion of optimal solutions of CND in Waxman networks are missing due to its



A Waxman networks B Power-law networks C Terrorist network

Figure 4-6. The performance evaluation of HILPR against the degree and betweenness centrality algorithms for the CLD problem



A Waxman networks B Power-law networks C Terrorist network

Figure 4-7. The performance evaluation of HILPR against the degree and betweenness centrality, and CNLS algorithms for the CND problem

extremely high computational complexity, which is because the network is neither almost intact nor almost fragmented). The pairwise connectivity derived from degree centrality algorithms is much worse than HILPR algorithm because the links or nodes of higher degrees could already connect other critical links or nodes and therefore are not necessary to be counted as critical any more. For instance, the hub nodes (nodes of high degree) in power-law networks are not necessarily connected with each other such that the removal of two hub nodes could be less effective to reduce the pairwise connectivity than the removal of two other nodes which can disconnect the network. The betweenness centrality performs worst due to the lack of all paths information rather than only shortest paths. That is, a pair of nodes can still be connected even when only the shortest path between them is destroyed. The reason why our HILPR algorithm outperforms CNLS is mainly because of the different strategy to choose the initial critical nodes before doing the local search. The CNLS method is only based on the maximum

degree from a maximal independent set, hoping that the removal of these nodes can greatly fragment the network. However, this is not always true since many nodes in the maximal independent set are usually of low degree, and consequently do not play an important role in destroying the network. In our HILPR approach, by solving the LP and rounding the top elements iteratively, we take into account all possible paths and connections between different nodes such that the critical elements can be accurately identified.

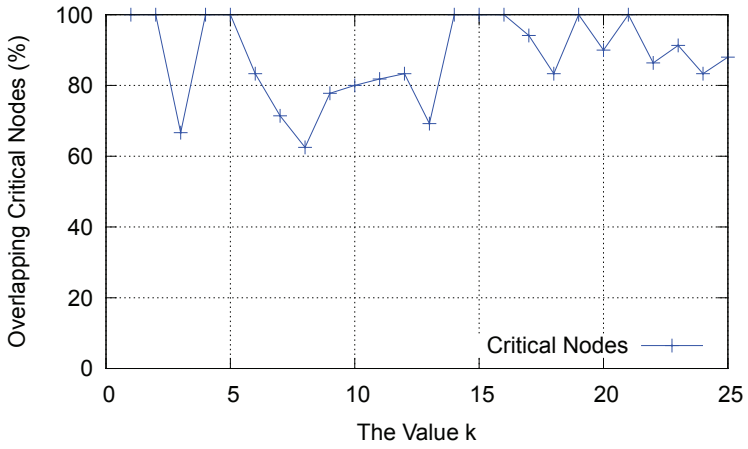


Figure 4-8. Overlapping critical nodes between optimal solution and HILPR in terrorist network

Specifically, in order to further show the effectiveness of our metric and algorithm, i.e., the critical links and nodes in real-world networks can be correctly detected using our algorithm, we dig into the real terrorist network in which the identities of nodes are available. The results returned by our HILPR algorithm show that we can detect the real important personnel by minimizing total pairwise connectivity. For instance, the two nodes 37 and 48 in the terrorist network, which have been shown as the leaders in [57**?**], can be correctly detected using our HILPR algorithm as long as $k = 2$. Yet, if we use degree and betweenness centrality methods, only node 37 can be detected when $k = 2$ and node 48 will not be detected until k is chosen to be 6 and 5 respectively.

Even though our objective is to minimize the pairwise connectivity, we are still interested in the overlapping percentage of critical nodes our HILPR algorithm returns and the optimal critical nodes. As reported in Fig. 4-8, in the real terrorist network, optimal critical nodes can be 100% successfully detected using our HILPR algorithm in more than 1/3 cases for different k values. The average overlapping percentage is around 80% since there exist some nodes playing the same role in network connectivity such that the pairwise connectivity still can be minimized although our HILPR algorithm identifies different critical nodes from the optimal solution.

Moreover, the running time of our HILPR algorithm is less than 5 seconds in all these three networks, for detecting either critical links or nodes, which is only slightly worse than centrality algorithms (1-2 seconds) and CNLS algorithm (2-3 seconds). Especially when k is small, i.e., only the most critical elements are required to be detected, our algorithm can finish around 3 seconds, which further illustrates the effectiveness of our HILPR algorithm in terms of both solution quality and running time.

Metric Evaluation

We evaluate the residual network obtained by HILPR algorithm under various network vulnerability metrics. As has been shown in Fig. 4-6 and 4-7, degree centrality and betweenness centrality cannot accurately reflect the network vulnerability. Therefore, we focus on the following three other metrics: (1) *average shortest path length* (ASP) between each node-pairs (the shortest distance is 0 if the pair of nodes are not connected), (2) *average available flows* (AAF) between each node-pairs, and (3) *global clustering coefficients* (GCC) defined as $\frac{\#\text{closed triplets}}{\#\text{connected triples of vertices}}$, in which a closed triplet consists of three nodes that are connected by three undirected ties.

Particularly, we are interested to see how the values of these metrics change in the residual network after we remove the critical elements which can successfully reduce the pairwise connectivity of the network. Since our HILPR algorithm can successfully detect the real critical links and nodes as discussed in the previous subsection, we

confidently evaluate the above three metrics on the residual graphs obtained by HILPR algorithm. Fig. 4-9 shows the changes in values of the above three metrics after removing different number of critical links or nodes. Unfortunately, none of these three metrics in residual networks can clearly cast the network vulnerability. As for the ASP, we can only consider the ASP within each connected component after the network is disconnected; otherwise the ASP becomes infinite and therefore fails to measure the network vulnerability. However, the value of ASP within connected components is either irregular (Fig. 4-9A) or contrary to the intuition, i.e., ASP usually increases with after removing critical elements (Fig. 4-9B). Similarly, the AAF fails to assess the network vulnerability due to its irregularity for critical links. The monotonous decease of AAF in the residual networks after removing critical nodes is greatly due to the disconnection of the network, which reduces the flow from two nodes in different connected components to 0. However, the nodes disconnecting the network are not necessary to be critical nodes. At last, the variation of GCC values is irregular for both critical links and nodes due to the simultaneous decrease of the number of connected triples of vertices. Particularly, when the network is highly fragmented, this metric can easily become infinite and meaningless (in the residual network of $k \geq 22$ as shown in Fig. 4-9B) since the number of connected triples of vertices becomes 0.

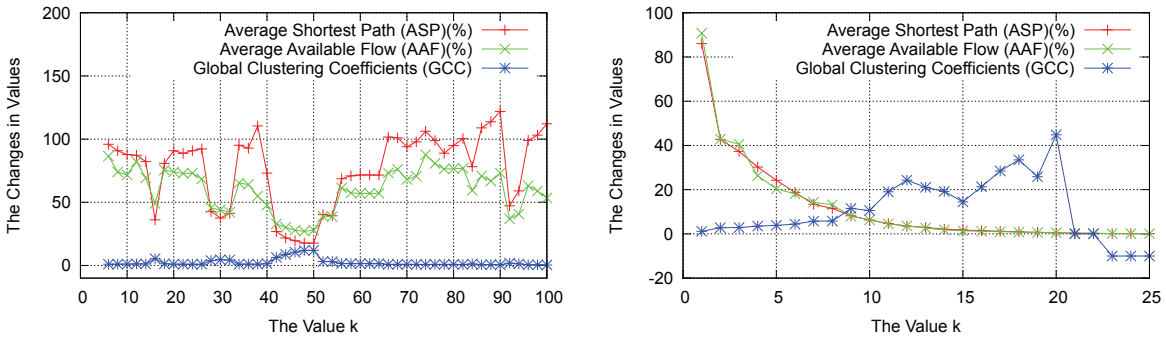


Figure 4-9. The comparison of different metrics on terrorist network

4.2.3 TRGA Approach under Cascading Failures

As illustrated in Section 3.7, the cascading failures could lead to the entire different set of critical elements. In this subsection, we propose our solution, a *Traceback and LP Rounding-Greedy Algorithm* (TRGA) for solving the CCND problem.

4.2.3.1 TRGA: an iterative 2-phase algorithm

In a big picture, TRGA algorithm (Algorithm 9) iteratively detects the most vulnerable nodes until k most vulnerable nodes, in which each iteration is two-fold: (1) identifying the ultimate failure nodes after cascading failures; (2) tracing back the vulnerable nodes based on the above failure nodes. In addition, TRGA also takes into account the lazy-update and constraint pruning techniques in each iteration further reduce its time complexity. In the end, a local search is provided to improve the solution quality. The rest of this subsection discusses two steps in each iteration in detail.

Phase 1: Ultimate Failure Nodes Identification In order to detect the ultimate failure nodes, the idea is to first guess the extent of fragmentation in residual networks, i.e., the number of connected node-pairs at last, and then identify these nodes based on the iterative rounding approach in [77], which has been show to be one of the best approaches for detecting critical nodes when failures are not cascaded. Denoting the residual pairwise connectivity as $\mathfrak{P}$, we estimate it based on the following intuition and observation: the larger the degree, the more vulnerable the node is in a network [84]. Therefore, in each iteration, we choose the k' highest degrees in the residual network ($k' = k -$ #detected vulnerable nodes) to simulate the cascading failures after deleting them and obtain the pairwise connectivity $\mathfrak{P}$. Then, we have the following Integer Linear

Programming (ILP) formulation:

$$\begin{aligned} \min \quad & \sum_{i \in V} v_i \\ \text{s.t.} \quad & v_i + v_j + u_{ij} \geq 1 && \forall (i,j) \in E \\ & u_{ij} + u_{jh} - u_{hi} \leq 1 && \forall i,j,h \in V \\ & \sum_{i,j \in V} u_{ij} \leq \mathfrak{P} \\ & v_i \in \{0,1\}, u_{ij} \in \{0,1\} \end{aligned} \tag{4–5}$$

where v_i is further defined as

$$v_i = \begin{cases} 1, & \text{if node } i \text{ is deleted (i.e., vulnerable nodes)} \\ 0, & \text{otherwise} \end{cases}$$

and

$$u_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}$$

The first additional constraint guarantees that at least one endpoint of a link has to be deleted if its two endpoints are disconnected in the optimal solution. The second constraint imposes the triangular connectivity while the third constraint means that the total pairwise connectivity after cascading failures has to be at most $\mathfrak{P}$. To solve it effectively, we borrow the idea in [77] to relax the equation (4–5), iteratively solve the LP and round the largest v_i. Likewise, we further apply constraints pruning in solving the LP and local search at the end of the whole ultimate failure nodes identification.

Phase 2: Vulnerable Nodes Tracing Back With the set of ultimate failure nodes after cascading failures, we trace back to the vulnerable nodes based on the following greedy algorithm. In particular, in each iteration, we select a node which can lead to the collapse of most ultimate failure nodes, call *cascading influence*, by simulating the failure cascades after removing each node.

To obtain the cascading influence of each node at each iteration, the easiest approach is to recompute for each node, yet this approach is extremely time-consuming. Instead, we apply the lazy-update process after the initial failure influence. Specifically, after the simulation at the first iteration, we maintain a max priority queue Q in which the priority is their cascading influence. In each iteration, the node u with the highest cascading influence is extracted and we recompute the extra nodes needed to fail u. In the next iteration, u will be selected if it still has the highest priority. Otherwise, u is pushed back to the priority queue, meanwhile the new node with the highest priority will be picked. Note that if the number of selected vulnerable nodes are larger than k', we will choose the k' highest degrees in the residual network from previous round instead and move on to the local search phase directly.

4.2.3.2 Optimality of CCND problem

In this subsection, we propose the following Integer Linear Programming (ILP) formulation for CCND problem in order to obtain its optimal solution. Next, we apply a sparse metric technique to further reduce the number of constraints, meanwhile keep the same optimal result.

Mathematical Formulation

For each pair of nodes $i, j \in V$, we define an indicator variable u_{ij} as:

$$u_{ij} = \begin{cases} 1, & \text{if } i \text{ and } j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}$$

and for all integers $t \in [0, d]$, we define

$$v_i^t = \begin{cases} 1, & \text{if node } i \text{ fails in round } t \\ 0, & \text{otherwise} \end{cases}$$

Note that $v_i^0 = 1$ when node i is a vulnerable node and fails at the beginning. Then we have the following ILP:

$$
\begin{aligned}
\min \quad & \sum_{i,j \in V} u_{ij} \\
\text{s.t.} \quad & v_i^d + v_j^d + u_{ij} \geq 1 && \forall (i,j) \in E \\
& u_{ij} + u_{jh} - u_{hi} \leq 1 && \forall i,j,h \in V \\
& \sum_{i \in V} v_i^0 \leq k \\
& \sum_{j \in N(v_i)} v_j^{t-1} + \theta \cdot deg(v_i) v_i^{t-1} \\
& \geq \theta \cdot deg(v_i) v_i^t && \forall i \in V, \forall 0 \leq t \leq d \\
& v_i^t \geq v_i^{t-1} && \forall 0 \leq t \leq d \\
& \forall i \in V, 0 \leq t \leq d \\
& v_i^t \in \{0,1\} && \forall 0 \leq t \leq d \\
& u_{ij} \in \{0,1\}
\end{aligned}
\tag{4–6}
$$

where the objective is to minimize the total pairwise connectivity. The first constraint guarantees that at least one endpoint of a link has to be deleted after d round cascades if its two endpoints are disconnected in the optimal solution. The second constraint imposes the triangular connectivity. That is, if node i and j are connected, node j and h are connected, node i and h have to be connected. The third constraint means that the total pairwise connectivity after d round failure cascades is at most β fraction of all node-pairs. The last two constraints deals with the cascades process and keeps failed nodes to be failure in the following rounds respectively.

4.2.3.3 Experimental evaluation

In this section, we evaluate the performance of our TRGA algorithm on different types of synthetic and real-world networks. The simulation is implemented using the CPLEX optimization suite from ILOG, which includes the simplex method [42], the branch & bound algorithm, and advanced cutting-plane techniques [86].

The three networks we use to evaluate the performance of our proposed TRGA algorithm are described as follows: (1) US Network Assets compiled by [77] with 71 nodes and 98 edges, which provides the current customer needs in XO Communications service. This experiment attempts to evaluate the performance of TRGA on a real-world communication network. In order to maintain the functionality of this communication network, we need to protect the most critical ISPs corresponding to the vulnerable nodes identified by TRGA; (2) Power-law network topology generated by igraph library [26] using the model in [2], with $\beta = 1.8$ and 70 nodes; (3) Small-world network topology generated by igraph library [26] using Watts and Strogatz model in [70], with $k = 2$, $\mu = 0.2$ and 70 nodes. The selection of parameters in these two synthetic networks is to keep the similar density as the US Network Assets network and also show the comparison with optimal solutions. We generate 100 instances for both power-law and small-world networks and show the average results.

In order to show the effectiveness of our proposed TRGA algorithm, we compare it with the optimal solution obtained by solving the ILP(4–6) directly. We also compare TRGA with two centrality approaches: degree centrality (DC) and betweenness centrality (BC), which are often used in network analysis [17]. In DC, the k nodes of largest degrees are selected as vulnerable nodes, and in BC, the k and nodes with largest betweenness are selected as vulnerable nodes obtained using [19], where the betweenness of a node is defined as the number of shortest paths among all pairs of nodes that passes through it.

Fig. 4-10 reports the comparison of the above TRGA algorithm and centrality algorithms for CCND on the above three different networks. In these figures, we notice that the solution of TRGA algorithm is very closely approaching the optimal solution for both CCND on all these three networks. Except in power-law networks in which nodes of high degrees have been shown as important nodes [84], the pairwise connectivity derived from degree centrality algorithms is much worse than TRGA algorithm especially

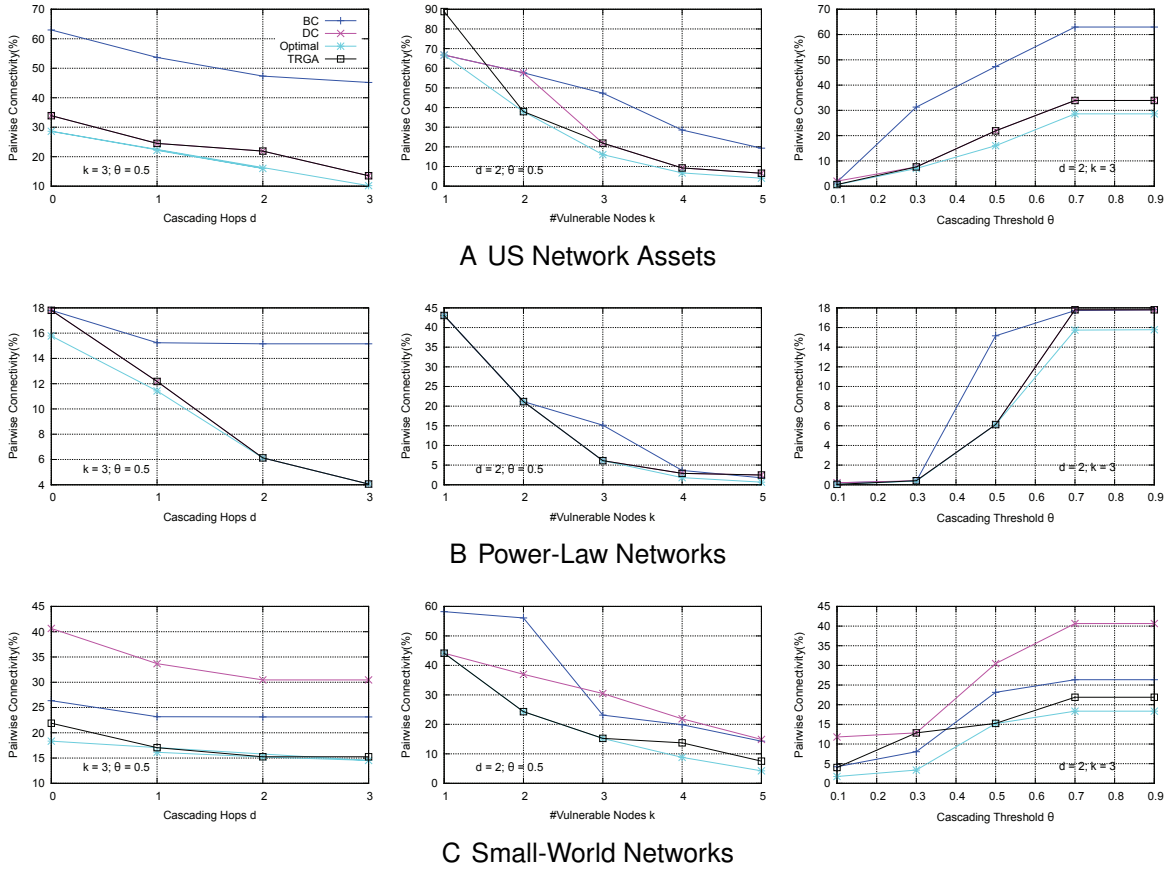


Figure 4-10. The performance evaluation of TRGA against degree and betweenness centrality algorithms for the CCND problem

in small-world networks due to their homogeneity in node degrees such that the nodes of higher degrees could already connect other vulnerable nodes and therefore are not necessary to be counted as vulnerable any more. The betweenness centrality performs worst in both power-law networks and US Network Assets due to the lack of all paths information rather than only shortest paths. That is, a pair of nodes can still be connected even when only the shortest path between them is destroyed. Yet, it outperforms degree centrality in small-world networks, in which the difference of degrees is not substantial. In our TRGA approach, in the first phase of each iteration, by solving the LP and rounding the top elements iteratively, we take into account all possible paths and connections between different nodes such that the critical elements can be accurately identified. Meanwhile, the second phase in TRGA with the

back-tracing can also precisely detect the original vulnerable nodes by providing more information than only degree or betweenness. Moreover, the running time of our TRGA algorithm is less than 10 seconds (due to the LP solver) in all these three networks, for detecting vulnerable nodes, which is acceptable compared with centrality algorithms (1-2 seconds), and over 100 times faster than obtaining optimal solution even with sparse metric. Besides, US Networks Assets as a well-designed communication network in practice, even with lower density, is shown to be the most robust among these three topologies.

4.3 Related Works

Many existing works on network vulnerability assessment mainly focus on the centrality measurements [17], including degree, betweenness and closeness centralities, average shortest path length [3], global clustering coefficients [60].

Due to the failures to assess the network vulnerability using above measurements, Sun *et al.* [80] first proposed the total pairwise connectivity as an effective measurement and empirically evaluate the vulnerability of wireless multihop networks using this metric. Arulselvan *et al.* [9] showed the challenge of CND problem by proving its NP-completeness. Later on, the β-disruptor problem was defined by Dinh *et al.* [27] to find a minimum set of links or nodes whose removal degrades the total pairwise connectivity to a desired degree. They proved the NP-completeness of this problem with respect to both links and nodes and the corresponding inapproximability results. Even for the tree topology, Di Summa *et al.* [61] found that the discovery of critical nodes also remains NP-complete using this metric. In this paper, we further investigate the theoretical hardness of both CLD and CND on UDGs and PLGs.

In addition, there are a few effective solutions in the literature of the network vulnerability assessment based on the pairwise connectivity. Arulselvan *et al.* [9] designed a heuristic (CNLS) to detect critical nodes, which is however still far away from the optimal solution in large-scale and dense networks. In [27], Dinh *et al.* proposed

pseudo-approximation algorithms to solve the β-disruptor problem. However, this problem is defined differently than ours and hard to use its solution when we only know the available cost to destroy or protect these critical links or nodes.

When failures are cascaded, these results are no longer valid, in which the vulnerability of networks could be substantially different. Most of the works regarding cascading failures mainly focus on models [25, 45, 85]. Moreover, there are some other papers providing some experimental analysis [29, 67]. Unfortunately, the theoretical works are lacked, which are crucial to the network design and proactive protection. Therefore, we provide a probabilistic analysis to assess the vulnerability for complex networks in the case of cascading failures, leading to deep insights to the robustness of various networks under random failures.

In addition, most of works on network vulnerability assessment for adversarial attacks are also studied without taking into account the cascading failures. Besides the widely-used centrality measurements [3, 17, 60], Arulselvan *et al.* [9] first proposed the total pairwise connectivity as an effective measurement, based on which they propose the CND problem and designed a heuristic to detect critical nodes. The β-disruptor problem was later defined by Dinh *et al.* [27] followed by pseudo-approximation algorithms. Unfortunately, these approaches fail to accurately identify the vulnerable nodes in the presence of cascading failures. In this paper, we further investigate the theoretical hardness the CVND problem, along with an effective algorithm.

Algorithm 9: TRGA Algorithm

Input : Network G, Threshold θ
Output: The set of k vulnerable nodes S

$k' \leftarrow k$;
$S \leftarrow \emptyset$;
while $|S| < k$ **do**
  $k' \leftarrow k - |S|$;
  $D \leftarrow k'$ largest degree nodes in $G[V \setminus S]$;
  $\mathfrak{P} \leftarrow$ #failed nodes after cascading failures by removing D from $G[V \setminus S]$;
  $U \leftarrow \emptyset$;
  `// Ultimate Failure Nodes Identification`
  while *Pairwise Connectivity* $> \mathfrak{P}$ **do**
    Use Constraint Pruning in [77] to solve the LP formulation with $\mathfrak{P}$;
    $\mathfrak{P} \leftarrow \mathfrak{P}-$ disconnected node-pairs after removing u;
    $u \leftarrow$ the node with largest v_i^*;
    $U \leftarrow U \cup \{u\}$;
    $G \leftarrow G[V \setminus \{u\}]$;
  end
  `// Vulnerable Nodes Tracing Back`
  $Q \leftarrow \emptyset$; `// Priority Queue`
  $S' \leftarrow \emptyset$;
  while $\exists$ *one node does not fail* **do**
    if $Q = \emptyset$ **then**
      foreach *node* u **do**
        Calculate the cascading influence after removing u from G;
      end
      Construct Q based on cascading influence of each node;
    end
    else
      $S' \leftarrow S' \cup$ the node in Q with max priority;
      Update cascading influence caused by removing this node;
    end
  end
  if $|S'| > k'$ **then**
    $S \leftarrow S \cup k'$ largest degree nodes in $G[V \setminus S]$;
  end
  else
    $S \leftarrow S \cup S'$;
  end
end
`// Local Search`
$S^* \leftarrow S$;
foreach *node* $u \in S$ **do**
  Swapping(u); (Algorithm 2 in [77] by replacing $f(G, S')$ with the pairwise connectivity function of residual graph G after removing S');
end
$S \leftarrow S^*$;
return S;

CHAPTER 5
CONCLUSION

In this dissertation, we first analyzed the approximation hardness and inapproximability of optimal substructure problems on power-law graphs. These problems are only illustrated in the literature not be able to approximated into some constant factors on both general and simple power-law graphs although they remain APX-hard. On the contrary, we also show that Max Clique and Graph Coloring are still very hard to be approximated since the optimal solutions to these problems are dependent on the structure of local graph component rather than global graph. In other words, the power-law distribution in degree sequence does not help much for such optimization problems without the property of optimal substructure. Moreover, we proposed a algorithm framework, along with a theoretical framework for analyzing approximation ratios, based on the idea of percolating the power-law graph from the nodes of lowest degree to other nodes.

In addition, we study the robustness of power-law networks under various threats, i.e. random failures, preferential attacks and degree-centrality attacks. Essentially, the power-law networks are illustrated to extremely tolerate random failures. In the meanwhile, they are more robust under both preferential attacks and degree-centrality attacks if they have a smaller exponential factor β. When failures can be cascaded, we showed that power-law networks are extremely vulnerable even with very small β.

In order to provide an optimal design of power-law networks, we further exploit the topologies of practical real-world networks by optimizing the costs and guaranteeing their robustness. The best range of the exponential factor β is illustrated to be $[1.8, 2.5]$, which gives a reasonable explanation for the topologies of most real-world networks. When $\beta < 1.8$, the network maintenance cost is very expensive, and when $\beta > 2.5$, the network robustness is unpredictable since it depends on the specific attacking strategy. Also, we study CLD and CND optimization problems to identify critical links and nodes in

a network whose removals maximally destroy the network's functions. We proved their NP-hardness and proposed HILPR, a novel LP-based rounding algorithm, for efficiently solving CLD and CND problems in a timely manner. In the present of cascading failures, we further study CCND problem and developed the effective iterative 2-phase TRPA algorithm. The experiments on various synthetic and real-world networks illustrated the good performance of our proposed approaches.

REFERENCES

[1] "http://networkx.github.io/documentation/latest/reference/generators.html." 1998.

[2] Aiello, William, Chung, Fan, and Lu, Linyuan. "A Random Graph Model for Power Law Graphs." *Experimental Math* 10 (2001): 53–66.

[3] Albert, R., Albert, I., and Nakarado, G. L. "Structural vulnerability of the North American power grid." *Phys. Rev. E* 69 (2004).2: 025103.

[4] Albert, R., Jeong, H., and Barabasi, A. L. "The diameter of the world wide web." *Nature* 401 (1999): 130–131.

[5] ———. "Error and attack tolerance of complex networks." *Nature* 406 (2000).6794: 378–382.

[6] Alderson, David, Doyle, John, Govindan, Ramesh, and Willinger, Walter. "Toward an optimization-driven framework for designing and generating realistic Internet topologies." *SIGCOMM Comput. Commun. Rev.* 33 (2003).1: 41–46.

URL http://doi.acm.org/10.1145/774763.774769

[7] Alderson, David L. and Doyle, John C. "Contrasting views of complexity and their implications for network-centric infrastructures." *Trans. Sys. Man Cyber. Part A* 40 (2010).4: 839–852.

URL http://dx.doi.org/10.1109/TSMCA.2010.2048027

[8] Alimonti, Paola and Kann, Viggo. "Hardness of Approximating Problems on Cubic Graphs." *CIAC '97*. London, UK: Springer-Verlag, 1997, 288–298.

[9] Arulselvan, Ashwin, Commander, Clayton W., Elefteriadou, Lily, and Pardalos, Panos M. "Detecting critical nodes in sparse graphs." *Comput. Oper. Res.* 36 (2009): 2193–2200.

URL http://dx.doi.org/10.1016/j.cor.2008.08.016

[10] Austrin, Per, Khot, Subhash, and Safra, Muli. "Inapproximability of Vertex Cover and Independent Set in Bounded Degree Graphs." *CCC '09*. 2009, 74–80.

[11] Barabasi, A. L. and Albert, R. "Emergence of scaling in random networks." *Science (New York, N.Y.)* 286 (1999).5439: 509–512.

URL http://view.ncbi.nlm.nih.gov/pubmed/10521342

[12] Barabási, AL. "Emergence of Scaling in Complex Networks." *Handbook of Graphs and Networks* (2003).

[13] Bianconi, G. and Barabási, A. L. "Bose-Einstein condensation in complex networks." 2000.

URL http://arxiv.org/abs/cond-mat/0011224

[14] Bollobas, B. *Random Graphs*. Cambridge University Press, 2001.

[15] Bollobás, Béla, Riordan, Oliver, Spencer, Joel, and Tusnády, Gábor. "The degree sequence of a scale-free random graph process." *Random Struct. Algorithms* 18 (2001).3: 279–290.

URL http://dx.doi.org/10.1002/rsa.1009

[16] Bondy, J.A. and Murty, U.S.R. *Graph theory with applications*. MacMillan London, 1976.

[17] Borgatti, Stephen P. and Everett, Martin G. "A Graph-theoretic perspective on centrality." *Social Networks* 28 (2006).4: 466 – 484.

URL http://www.sciencedirect.com/science/article/B6VD1-4J32JGJ-1/2/c8bf3911c0e5bb0e701c0a541e380475

[18] Bornholdt, Stefan and Schuster, Heinz Georg, eds. *Handbook of Graphs and Networks: From the Genome to the Internet*. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[19] Brandes, Ulrik. "A Faster Algorithm for Betweenness Centrality." *Journal of Mathematical Sociology* 25 (2001): 163–177.

[20] Chlebík, M. and Chlebíková, J. "Approximation hardness of dominating set problems in bounded degree graphs." *Inf. Comput.* 206 (2008).11: 1264–1275.

[21] Chung, Fan and Lu, Linyuan. "Connected Components in Random Graphs with Given Expected Degree Sequences." *Annals of Combinatorics* 6 (2002).2: 125–145.

URL http://dx.doi.org/10.1007/PL00012580

[22] ———. "Concentration Inequalities and Martingale Inequalities: A Survey." *Internet Mathematics* 3 (2006).1: 79–127.

URL http://dx.doi.org/10.1080/15427951.2006.10129115

[23] Cohen, Reuven, Erez, Keren, Ben-Avraham, Daniel, and Havlin, Shlomo. "Resilience of the Internet to Random Breakdowns." *Physical Review Letters* 85 (2000).21: 4626+.

URL http://dx.doi.org/10.1103/PhysRevLett.85.4626

[24] Cooper, Colin and Frieze, Alan. "A general model of web graphs." *Random Struct. Algorithms* 22 (2003).3: 311–335.

URL http://dx.doi.org/10.1002/rsa.10084

[25] Crucitti, P., Latora, V., and Marchiori, M. "Model for cascading failures in complex networks." *Phys Rev E Stat Nonlin Soft Matter Phys* 69 (2004).4 Pt 2: 045104.

[26] Csardi, Gabor and Nepusz, Tamas. "The igraph software package for complex network research." *InterJournal* Complex Systems (2006): 1695.

URL http://igraph.sf.net

[27] Dinh, T.N., Xuan, Ying, Thai, M.T., Pardalos, P.M., and Znati, T. "On New Approaches of Assessing Network Vulnerability: Hardness and Approximation." *Networking, IEEE/ACM Transactions on* 20 (2012).2: 609 –619.

[28] Dinur, Irit and Safra, Samuel. "On the Hardness of Approximating Minimum Vertex Cover." *Annals of Mathematics* 162 (2004): 2005.

[29] Dobson, I., Carreras, B.A., Lynch, V.E., and Newman, D.E. "Complex systems analysis of series of blackouts: Cascading failure, critical points, and self-organization." *Chaos* 17 (2007).2: 026103.

[30] Doyle, J. C., Alderson, D. L., Li, L., Low, S., Roughan, M., Shalunov, S., Tanaka, R., and Willinger, W. "The "robust yet fragile" nature of the Internet." *Proc. Natl. Acad. Sci. USA* 102 (2005).

[31] Erdos, P. and Gallai, T. "Graphs with prescribed degrees of vertices." *Mat. Lapok* 11 (1960): 264–274.

[32] Eubank, Stephen, Kumar, V. S. Anil, Marathe, Madhav V., Srinivasan, Aravind, and Wang, Nan. "Structural and algorithmic aspects of massive social networks." *SODA '04*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2004, 718–727.

[33] Fabrikant, Alex, Koutsoupias, Elias, and Papadimitriou, Christos H. "Heuristically Optimized Trade-Offs: A New Paradigm for Power Laws in the Internet." *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*. ICALP '02. London, UK, UK: Springer-Verlag, 2002, 110–122.

URL http://dl.acm.org/citation.cfm?id=646255.684438

[34] Faloutsos, Michalis, Faloutsos, Petros, and Faloutsos, Christos. "On power-law relationships of the Internet topology." *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*. SIGCOMM '99. New York, NY, USA: ACM, 1999, 251–262.

URL http://doi.acm.org/10.1145/316188.316229

[35] Ferrante, Alessandro, Pandurangan, Gopal, and Park, Kihong. "On the hardness of optimization in power-law graphs." *Theoretical Computer Science* 393 (2008).1-3: 220–230.

[36] Gkantsidis, Christos, Mihail, Milena, and Saberi, Amin. "Conductance and congestion in power law graphs." *SIGMETRICS Perform. Eval. Rev.* 31 (2003).1: 148–159.

[37] Gkantsidis, Christos, Mihail, Milena, and Zegura, Ellen. "The Markov Chain Simulation Method for Generating Connected Power Law Random Graphs." *In Proc. 5th Workshop on Algorithm Engineering and Experiments (ALENEX). SIAM*. 2003.

[38] Glover, F. and Laguna, M. "Tabu Search." 1997.

[39] Goyal, Amit, Bonchi, Francesco, and Lakshmanan, Laks V.S. "Learning influence probabilities in social networks." *Proceedings of the third ACM WSDM'10*. WSDM '10. New York, NY, USA: ACM, 2010, 241–250.

URL http://doi.acm.org/10.1145/1718487.1718518

[40] Halldórsson, Magnús and Radhakrishnan, Jaikumar. "Greed is good: approximating independent sets in sparse and bounded-degree graphs." *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*. STOC '94. New York, NY, USA: ACM, 1994, 439–448.

URL http://doi.acm.org/10.1145/195058.195221

[41] Hastad, J. "Clique is hard to approximate within $n^{1-\epsilon}$." *FOCS '96*. Washington, DC, USA: IEEE Computer Society, 1996, 627.

[42] Hillier, Frederick S. and Lieberman, Gerald J. *Introduction to operations research, 4th ed.* San Francisco, CA, USA: Holden-Day, Inc., 1986.

[43] Holme, P., Kim, B. J., Yoon, C. N., and Han, S. K. "Attack vulnerability of complex networks." *Phys. Rev. E* 65 (2002).5: 056109.

[44] Hopcroft, J. E. and Karp, R. M. "An $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs." *SIAM J. Comput.* 2 (1973): 225–231.

[45] Iyer, S.M., Nakayama, M.K., and Gerbessiotis, A.V. "A Markovian Dependability Model with Cascading Failures." *Computers, IEEE Transactions on* 58 (2009).9: 1238 –1249.

[46] Jamakovic, A. and Uhlig, S. "On the relationship between the algebraic connectivity and graph's robustness to node and link failures." *Next Generation Internet Networks, 3rd EuroNGI Conference on*. 2007, 96–102.

[47] Jamakovic, A. and Van Mieghem, P. "On the robustness of complex networks by using the algebraic connectivity." *Proceedings of the 7th international IFIP-TC6 networking conference on AdHoc and sensor networks, wireless networks, next generation internet*. NETWORKING'08. Berlin, Heidelberg: Springer-Verlag, 2008, 183–194.

URL http://dl.acm.org/citation.cfm?id=1792514.1792537

[48] Janson, Svante, Łuczak, Tomasz, and Norros, Ilkka. "Large cliques in a power-law random graph." 2009. Comment: 13 pages.

URL http://arxiv.org/abs/0905.0561

[49] Kõnig, Dénes. "Gráfok és mátrixok." *Matematikai és Fizikai Lapok* 38 (1931): 116–119.

[50] Kaiser, M. and Hilgetag, C. C. "Edge vulnerability in neural and metabolic networks." *Biological Cybernetics* 90 (2004): 311–317. 10.1007/s00422-004-0479-1.

URL http://dx.doi.org/10.1007/s00422-004-0479-1

[51] Kann, Viggo. *On the Approximability of NP-complete Optimization Problems*. Ph.D. thesis, Royal Institute of Technology Stockholm, 1992.

[52] Karakostas, George. "A better approximation ratio for the vertex cover problem." *ACM Trans. Algorithms* 5 (2009).4: 41:1–41:8.

URL http://doi.acm.org/10.1145/1597036.1597045

[53] Keeling, M. J. "The effects of local spatial structure on epidemiological invasions." *Proc. R. Soc. B* 266 (1999).1421: 859–867.

[54] Kempe, David, Kleinberg, Jon, and Tardos, Eva. "Maximizing the Spread of Influence through a Social Network." *In KDD*. ACM Press, 2003, 137–146.

[55] ———. "Influential Nodes in a Diffusion Model for Social Networks." *IN ICALP*. Springer Verlag, 2005, 1127–1138.

[56] Kitsak, Maksim, Havlin, Shlomo, Paul, Gerald, Riccaboni, Massimo, Pammolli, Fabio, and Stanley, H. Eugene. "Betweenness centrality of fractal and nonfractal scale-free model networks and tests on real networks." *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* 75 (2007).5: 056115+.

URL http://dx.doi.org/10.1103/PhysRevE.75.056115

[57] Krebs, Valdis E. "Uncloaking Terrorist Networks." *First Monday* 7 (2002).4.

URL \protect\begingroup\catcode`\\active\def{}\catcode`%\active\let%%\let%%\catcode`#\active\def#{#}\def#{#}\catcode`\&12\relax\edef_{_}\let__\catcode`_\active\let__\let~~\let~~\let~~\let\\\edef${$}\edefREFERENCES{\endgroup\url@{http://firstmonday.org/issues/issue7_4/krebs/index.html}}REFERENCES

[58] Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E. "Stochastic models for the Web graph." *Proceedings of the 41st Annual*

Symposium on Foundations of Computer Science. FOCS '00. Washington, DC, USA: IEEE Computer Society, 2000, 57–.

URL http://dl.acm.org/citation.cfm?id=795666.796570

[59] Lakhina, Anukool, Papagiannaki, Konstantina, Crovella, Mark, Diot, Christophe, Kolaczyk, Eric D., and Taft, Nina. "Structural analysis of network traffic flows." *Proceedings of the joint international conference on Measurement and modeling of computer systems*. SIGMETRICS '04/Performance '04. New York, NY, USA: ACM, 2004, 61–72.

URL http://doi.acm.org/10.1145/1005686.1005697

[60] Luciano, Rodrigues, F.A., Travieso, G., and Boas, V. P. R. "Characterization of complex networks: A survey of measurements." *Advances in Physics* 56 (2007).1: 167–242.

URL http://dx.doi.org/10.1080/00018730601170527

[61] Marco Di Summa, Andrea Grosso. "Complexity of the Critical Node Problem over trees." *Optimization Online* (2011).

[62] Marin-Perianu, R. S., Scholten, J., Havinga, P. J. M., and Hartel, P. H. "Cluster-based service discovery for heterogeneous wireless sensor networks." *International Journal of Parallel, Emergent and Distributed Systems* 23 (2008).4: 325–346.

URL http://dx.doi.org/10.1080/17445760801930948

[63] Matisziw, T. C. and Murray, A. T. "Modeling s-t path availability to support disaster vulnerability assessment of network infrastructure." *Computers & Operations Research* 36 (2009).1: 16 – 26. Part Special Issue: Operations Research Approaches for Disaster Recovery Planning.

URL http://www.sciencedirect.com/science/article/B6VC5-4PP2D14-1/2/7e5b80edb520a880374876773c0a5e4d

[64] Medina, Alberto, Lakhina, Anukool, Matta, Ibrahim, and Byers, John. "BRITE: An Approach to Universal Topology Generation." 2001.

[65] Medina, Alberto, Matta, Ibrahim, and Byers, John. "On the Origin of Power Laws in Internet Topologies." Tech. rep., Boston University, Boston, MA, USA, 2000.

[66] Mihail, Milena and Papadimitriou, Christos H. "On the Eigenvalue Power Law." *RANDOM*. 2002, 254–262.

[67] Mirzasoleiman, Baharan, Babaei, Mahmoudreza, Jalili, Mahdi, and Safari, MohammadAli. "Cascaded failures in weighted networks." *Physical Review E* 84 (2011): 046114+.

URL http://dx.doi.org/10.1103/PhysRevE.84.046114

[68] Molloy, Michael and Reed, Bruce. "A critical point for random graphs with a given degree sequence." *Random Struct. Algorithms* 6 (1995): 161–179.

URL http://portal.acm.org/citation.cfm?id=259573.259582

[69] ———. "The Size of the Giant Component of a Random Graph with a Given Degree Sequence." *Comb. Probab. Comput.* 7 (1998).3: 295–305.

URL http://dx.doi.org/10.1017/S0963548398003526

[70] Newman, M. E. J. and Watts, D. J. "Renormalization group analysis of the small-world network model." *Physics Letters A* 263 (1999).4-6: 341–346.

[71] Norros, I. and Reittu, H. "On a conditionally Poissonian graph process." *Advances in Applied Probability* (2006): 38–59.

[72] P. Satorras, R. and Vespignani, A. "Immunization of complex networks." *Phys. Rev. E* 65 (2002).3: 036104.

[73] Pandurangan, Gopal. "https://sites.google.com/site/gopalpandurangan/papers-by-date." 2006.

[74] Redner, S. "How popular is your paper? An empirical study of the citation distribution." *The European Physical Journal B - Condensed Matter and Complex Systems* 4 (1998).2: 131–134.

URL http://dx.doi.org/10.1007/s100510050359

[75] Sachtjen, M. L., Carreras, B. A., and Lynch, V. E. "Disturbances in a power transmission system." *Physical Review E* 61 (2000).5: 4877–4882.

URL http://dx.doi.org/10.1103/PhysRevE.61.4877

[76] Serfling, R. J. "Probability Inequalities for the Sum in Sampling without Replacement." *The Annals of Statistics* 2 (1974).1: 39–48.

URL http://dx.doi.org/10.1214/aos/1176342611

[77] Shen, Y., Nguyen, N. P., Xuan, Y., and Thai, M. T. "On the Discovery of Critical Links and Nodes for Assessing Network Vulnerability." *Networking, IEEE/ACM Transactions on* PP (2012).99: 1.

[78] Shen, Yilin, Nguyen, Nam P., and Thai, My T. "Exploiting the Robustness on Power-Law Networks." *COCOON*. 2011, 379–390.

[79] Smirnov, Michael, Biersack, Ernst W., Blondia, Chris, Bonaventure, Olivier, Casals, Olga, Karlsson, Gunnar, Pavlou, George, Quoitin, Bruno, Roberts, James, Stavrakakis, Ioannis, Stiller, Burkhard, Trimintzios, Panos, and Mieghem, Piet Van,

eds. *Quality of Future Internet Services, COST Action 263 Final Report*, vol. 2856 of *Lecture Notes in Computer Science*. Springer, 2003.

[80] Sun, Fangting and Shayman, Mark A. "On pairwise connectivity of wireless multihop networks." *Int. J. Secur. Netw.* 2 (2007).1/2: 37–49.

URL http://dx.doi.org/10.1504/IJSN.2007.012823

[81] Sydney, A., Scoglio, C., Schumm, P., and Kooij, R. E. "Elasticity: topological characterization of robustness in complex networks." *Proceedings of the 3rd International Conference on Bio-Inspired Models of Network, Information and Computing Sytems*. BIONETICS '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, 19:1–19:8.

URL http://dl.acm.org/citation.cfm?id=1512504.1512529

[82] Tangmunarunkit, Hongsuda, Govindan, Ramesh, Jamin, Sugih, Shenker, Scott, and Willinger, Walter. "Network topology generators: degree-based vs. structural." *SIGCOMM*. 2002, 147–159.

[83] Vazirani, Vijay V. *Approximation algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[84] Wang, Xiao F. and Chen, Guanrong. "Complex networks: small-world, scale-free and beyond." *Circuits and Systems Magazine, IEEE* 3 (2003).1: 6–20.

URL http://dx.doi.org/10.1109/MCAS.2003.1228503

[85] Watts, Duncan J. "A simple model of global cascades on random networks." *Proceedings of the National Academy of Sciences* 99 (2002).9: 5766–5771.

URL http://www.pnas.org/content/99/9/5766.abstract

[86] Wolsey, L. *Integer Programming*. 1998.

[87] Yan, Guanhua, Chen, Guanling, Eidenbenz, Stephan, and Li, Nan. "Malware Propagation in Online Social Networks: Nature, Dynamics, and Defense Implications." *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security (AsiaCCS'11)*. Hongkong, China, 2011.

BIOGRAPHICAL SKETCH

Yilin Shen received his Ph.D. from the University of Florida in the spring of 2013 and his B.S. degree in applied mathematics from Donghua University, Shanghai, China, in 2005. His research focuses on vulnerability assessment and security of complex networks, including communication networks, wireless sensor networks and social networks, and designing approximation algorithms for network optimization problems. He is a student member of the IEEE.

CASCADING PROPAGATION AND OPTIMIZATION IN NETWORKS

By

DUNG T. NGUYEN

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2013

I dedicate this to the internal God

ACKNOWLEDGMENTS

I would like to thank my advisor Prof. My T. Thai for great advices during the PhD time. I was bestowed a precious gift working with her in four years. Her good examples are my treasure which have gradually made me stronger in both doing research and overcoming difficulties. It was lucky for me to get infected by her passion and curiosity when I was trying to solve challenging problems. When I was lacking of motivation, the ritual to get motivated may be quite simple: seeing her as an example.

I am thankful to Prof. Alireza Entezari, Prof. Tamer Kahveci, Prof. Sartaj K. Sahni, and Prof. J. Cole Smith for their time and constructive opinions.

I would like to thank all of my closed and acquainted friends for sharing their knowledge, perspectives, opinions, and enjoying moments.

TABLE OF CONTENTS

LIST OF TABLES

Table page

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

CASCADING PROPAGATION AND OPTIMIZATION IN NETWORKS

By

Dung T. Nguyen

August 2013

Chair: My T. Thai
Major: Computer Engineering

Cascading processes are more and more popular in highly connected networks. These processes are recognized in a wide range of networks with different contexts: the information diffusion in online social networks, the cascading crisis in the network of banks, the cascading failure in power networks, etc. Regardless of the network type and mechanism, they still share fundamental properties: (1) the root cause is the influences/dependencies between nodes of one or more networks, (2) the process often starts from a small group of nodes, (3) the impact is high due to the large number of involved nodes. It is thus crucial to study these processes and exploit them efficiently.

In this work, we study several optimization problems relating to the cascading process in networks. In particular, we mainly focus on two kinds of problems: (1) finding a small set of nodes which can maximize the impact through the cascading process and (2) finding a set of nodes with minimum size which causes the desired impact. Depending on the cascading mechanism, we design different strategies to solve the problem efficiently by exploiting both the properties of both the cascading mechanism and the network structure.

CHAPTER 1
INTRODUCTION

Nowadays, the world is more and more connected and we can see networks everywhere. Networks, from power substations in power networks, routers in communication networks, users in online social networks, etc., represent the interaction between entities and play crucial roles in the economy. Power networks are important infrastructure networks whose malfunction can lead the change or stopping of almost daily activities. On the other hand, large-scaled online social networks are easing the communication between people by providing a platform for users to connect and keep updating from each other. Due to the high impact of these networks on the economy, it is crucial to study phenomena which significantly affect activities in networks.

As entities in networks interact with each other, the cascading propagation is one of the most noticeable phenomena in networks. If an event happens at a particular entity, it can triggers events at other entities through the connection between entities in the network. For instance, the interaction between users in online social networks serves as the medium to spread information, ideas, and influences. Initially, only a small group of users aware of the information and share in the networks; then, the information is spread to their friends, friends of their friends, and so on. As a result, large number of users will aware of the information even before the mass media broadcast it as in the case of Michael Jackson's death [1]. In the power network, the cascading propagation can cause severe damage by multiplying the initial failure. In 2003, the initial failure of one power line triggered a series of failures which resulted in the outage of the majority of Italy [47]. The large-scaled effect of the cascading propagation inspires us to design methods to exploit its positive effects and prohibit negative effects. However, the cascading mechanisms are various in networks and too broad, thus we focus on investigating the cascading propagation in following settings:

1.1 Cascading Failure in a Network

Networks where the operation of a node strongly depends on the operation of other nodes like power networks are extremely vulnerable under cascading failure. In these networks, every node bears dynamic operational load depending on the demand. If the demand is high, the nodes' loads are high and may reach to their maximum operational capacities. In general, nodes share the network demand so that each of them can operate under the permitted capacity. However, when some nodes are malfunctioned or failed, they shift the load to nearby nodes in the network. These nodes may be forced to work beyond their capacities so they are overloaded redistribute their load onto other nodes. As a consequence, a large number of nodes may be overloaded and thereby the network halts the operation entirely.

As the failure of a small group of nodes may result in a catastrophic damage on the network operation, it is important to identify such groups. These nodes are critical to the operation of the network, thus we need to protect them from being attacked. Although existing literature provides various vulnerability assessment of networks under the cascading failure, there is still lacking of efficient solutions for this problem. These works mainly exploit the centrality measurement to locate most critical nodes which are not enough to capture the complicated interaction of nodes. We design new methods which consider both network structure and the interaction between nodes to provide better solutions.

1.2 Cascading Failure in Interdependent Networks

In reality, infrastructure networks are interdependent on each other at a large degree. The power stations in the power grid consume the fuel delivered by the transportation network to generate electricity and are controlled via the communication network. If the transportation network or the communication network encounters any problems, the operation of the power network will suffered. On the other hand, the power network provides electricity for routers of the communication network and electrical

vehicles. Therefore, the failure of a critical group of nodes in any network may result in a series of cascading failures across the system and cause catastrophic loss. If each network is treated separately, we will underestimate the vulnerability of networks.

We need to revisit the vulnerability assessment of networks taking into account the effect of interdependencies between networks. Let's consider the power and communication networks. In the attacking point of view, an attacker can analyze the interdependencies and identify nodes whose failures trigger a large-sized cascading process back and forth between networks. If we only investigate a single network, these nodes seem to be scot-free and we fail to protect them. Although there are many efficient methods to identify critical nodes in a single network, it is still lacking ones for interdependent networks. In this work, we propose a new centrality for interdependent networks which can be used to locate critical nodes efficiently.

1.3 Influence Diffusion in Multiple Online Social Networks

In the area of online social networks (OSNs), the cascading propagation of information transforms networks such as Facebook, Google+, and Twitter to a fruitful foundation for viral marketing. These networks equip users tools to connect and make new friends, to share opinions, to update information from friends, etc., thus attract a considerable fraction the population to join in. In return, users create the content and circulate the information at a level that has been achieved before by any of previous communication medium. In addition, users in online social networks also incur the same peer-pressure effect as the reality, i.e., in which an individual's opinion or decision is influenced by his friends and colleagues. These factors raise a practical important problem in OSNs: how to find the smallest set of influencers who can influence a massive number of users.

A noticeable property OSNs is the overlapping among major OSNs which has a strong impact on the diffusion of information. Since a user can share the information in all networks which he participates in, the influence of a user in all networks is significant

larger than in any network. Therefore, it is essential to evaluate the influence of users in multiple OSNs to identify most influential ones. However, we can not trivially mitigate evaluation methods for a single network to multiple networks. To overcome this difficulty, we propose novel schemes to couple multiple networks into one network reserving all diffusion information and solve the problem in the coupled network.

1.4 Organization

The rest of the work is organized as follows. Chapter 2 studies the vulnerability of power networks under the load redistribution model. In chapter 3, we present the cascading failure model for interdepent networks and propose algorithms to detect critical nodes. Next, chapter 4 investigates information diffusion in multiple online social networks. Finally, chapter 5 concludes the whole thesis.

CHAPTER 2
CASCADING FAILURE UNDER LOAD REDISTRIBUTION IN NETWORKS

The important role of power networks in the economy as well as in the society has attracted a great deal of research effort to analyze the vulnerability of these networks. The failure or malfunction of these networks can cause severe effect. On 28 September 2003 [47], the wide area blackout affected the major of Italy and made 3/4 of Italy without electricity for 2 hours, the traffic system is halted. This shows that large blackouts are not rared and can happen everywhere. Moreover, it implies that intentional attacks can cause mass damage to power networks. When the small number of components are attacked, the large blackout can happen in a very short time. Thus, it is crucial to identify most vulnerable components of the power network so that we can protect in advance.

The common denominator of large blackouts is that the failures of components happened according to the cascading manner. It often starts with the failure of one or a few components, then some other components are failed due to the dependencies with previous failed components. The failure of these components continue to cause other components fail. The process continues until there is no more failed component. The power statiosn which are nodes in the power network can only work well if the load is under the maximum capacity they can handle. When a station is overloaded, it can not work with the best performance or even fails. During the operation, the power network is designed such that all stations work under their capacity. But when some stations fails, other stations which are directly or indirectly connected with failed ones may have bear more load. If the load of a station surpasses its capacity, it will fail and continue to shred its load to other stations. As a result of the load redistribution process, a large number of failed stations may be failed at the end. We would like to predict the process so that we can prevent it, but the dependencies between stations make it difficult to do so. Thus, it

is necessary to develop efficient tools to analyze the sophisticated cascading process of failures in power networks.

In this chapter, we study the critical node detection problem in power networks under the load redistribution of nodes. Specifically, we aim to find the set of k nodes whose failures maximize the number failed nodes after the cascading failure. We design two efficient algorithms to solve problem: adaptive cascading potential algorithm and cooperating attack algorithm. The efficiency of each algorithm depends on the topological structure of the network, hence they compensate each other to solve the problem.

The rest of the chapter is organized as follows. We first present the load redistribution model and problem formulation in Section 2.1. Section 2.2 shows the hardness result. After that, we propose the cascading potential metric and design various algorithms in Section 2.3. We next introduce the cooperating algorithm which is efficient on robust networks in Section 2.4. Section 2.5 shows the experimental evaluation. Finally, we review the literature in Section 2.6 and summerize the chapter in Section 2.7.

2.1 Network Model and Problem Formulation

2.1.1 Graph Notations

The network is modeled by a weighted directed graph $G = (V, E)$ with vertex set V of $|V| = n$ vertices and edge set E of $|E| = m$ oriented connections between vertices. Each edge (u, v) is associated with a weight $w(u, v)$ presenting the operating parameter of the network. The higher $w(u, v)$ is, the more load is distributed from u to v. In addition, each vertex u has the current load $L(u)$ and a capacity $C(u)$. The capacity $C(u)$ is the maximum load that vertex u can accept. We denote the set of incoming neighbors, outgoing neighbors of u by N_u^- and N_u^+, respectively.

2.1.2 Cascading Failure Model

In the Load Redistribution model (LR-model) [56] [53], nodes are failed in the cascading manner due to the load redistribution of failed nodes. Initially, a set of nodes

S are failed, then the failures are propagated to other nodes in time steps. When node u fails, its load is redistributed to its neighbors as illustrated in Fig. 2-1. Each alive neighbor will received an additional load which is proportional to its weight. Precisely, each neighbor v of u will receive additional load:

$$\Delta L(v) = L(u) \times \frac{w(u,v)}{\sum_{z \in N_u^+} w(u,z)}$$

Due to the load redistribution, the load of some nodes are exceeding their capacities, hence fail in the next time step. The process of load redistribution and node failing will stop when there are no more failed nodes. The set of failed nodes caused by the initial failure of S is denoted by $F(S)$.

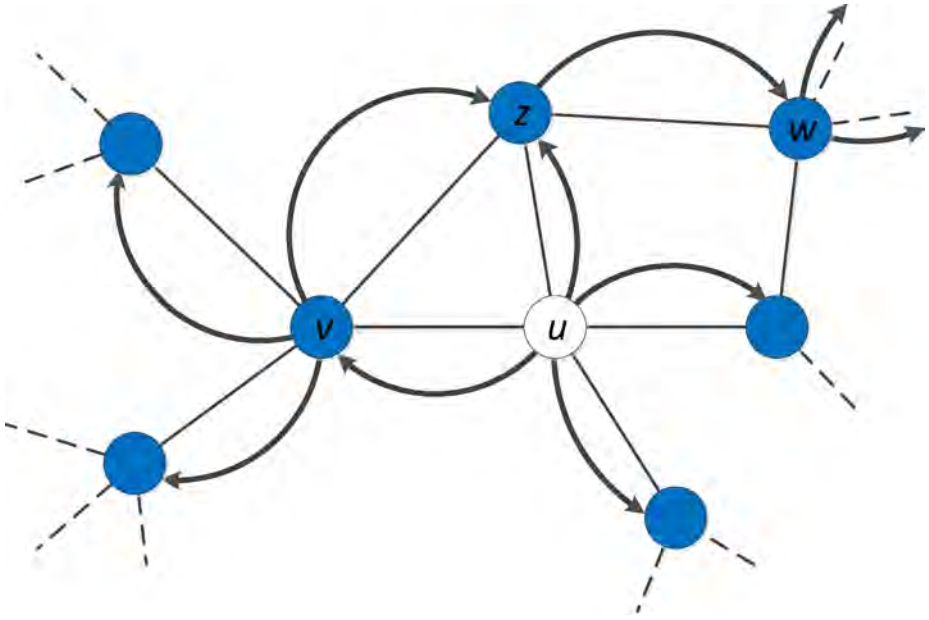


Figure 2-1. When node u fails, its load is redistributed to the neighbor nodes. Among these node, v receives a high portion of load from u and becomes overloaded. The load of v is redistributed to its neighbors which makes z fail. Finally, the load from z continues to cause w fail and the process stops.

2.1.3 Problem Definition

Due to the cascading failures, the failures of a small set of nodes S can result in a catastrophic number of failed nodes. These nodes becomes the target to attack the network. Additionally, given the same set of attacked nodes, different attacking orders lead to different outcomes. With the same attacking cost, the attacker can choose the best order with suitable time for each attacked node. However, the cascading failures happen very fast, it is almost impossible to schedule the failure of each node with specific time steps. We consider a more practical strategy in which target nodes are attacked one by one. The next node is taken down when the cascading process stops. In particular, given an order set $S = \{s_1, s_2, \dots, s_k\}$, the set of fails after s_i is attacked is $F_i(S) = F(F_{i-1}(S) \cup \{s_i\})$. Denote $F^+(S)$ as $F_k(S)$, the set of failed nodes when nodes in S is attacked serially. We formally define the problem as follows.

Definition 1 (Cascading Critical Node Problem (Cas-CNP))**.** *Given a network $G = (V, E)$ and an integer k, the problem asks to find a ordered subset $S \subseteq V$ of size $|S| = k$ such that the serial failures of nodes in S maximizes the number of failed nodes $F^+(S)$ under the LR-model.*

2.2 Inapproximability Result

In this section, we show the algorithmic hardness of the Cas-CN problem. We expect to design an algorithm that can identify the optimal seed set in an acceptable time. However, it may take the time as an exponential funtion of the number of nodes to compute even a set whose impact is close to the optimal set's. The hardness result is shown in Theorem 2.1.

Theorem 2.1. *It is NP-hard to approximate the CasCN problem within ratio of $O(n^{1-\epsilon})$ for any constant $1 > \epsilon > 0$.*

Proof. We use the gap-introduction reduction [51] to prove the inapproximability of the CasCN problem. Using a polynomial time reduction from Set Cover, to the CasCN problem, we show that if there exists a polynomial time algorithm that approximates the

later problem within $O(n^{1-\epsilon})$, then there exists a polynomial time algorithm to solve the former problem.

Definition 2 (Set Cover problem)**.** *Given a universe $\mathcal{U} = \{e_1, e_2, \dots, e_n\}$, a collection of subsets $\mathcal{S} = \{S_1, S_2, \dots, S_m\} \subseteq 2^{\mathcal{U}}$, and an integer k, the Set Cover problem asks whether or not there are k subsets whose union is $\mathcal{U}$.*

Instead of using the hardness result of the general Set Cover problem, we use the result on a restricted variant MIN3SC2 of the Set Cover problem where the sizes of subsets are at most 3 and each element appears in exactly two subsets.

Theorem 2.2 ([20])**.** *The Set Cover problem is NP-hard even when the sizes of subsets are bounded by 3 and each element appears in exactly two subsets.*

Reduction. Given an instance of the Set Cover problem $\mathcal{I} = (\mathcal{U}, \mathcal{S}, k)$ where each element appears in exactly two subsets, $n_1 = |\mathcal{U}|$ and $m_1 = |\mathcal{S}|$, we construct an instance $\mathcal{I}'$ of the CasCN problem as illustrated in Fig. 2-2.

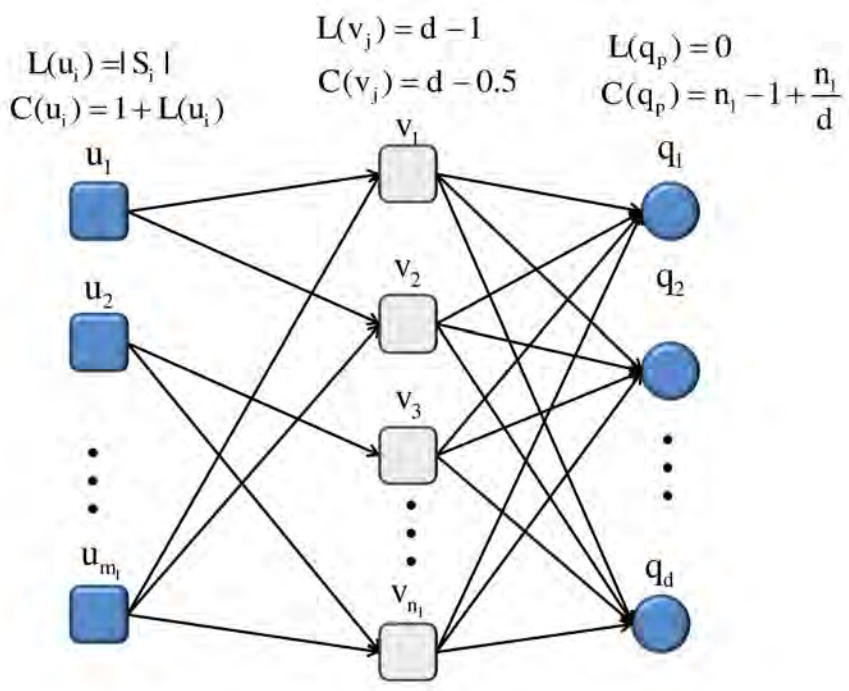


Figure 2-2. Reduction from MIN3SC2 to CasCN

The vertex set V. Add a *set vertex* u_i for each set $S_i \in \mathcal{S}$, an *element vertex* v_j for each element $e_j \in \mathcal{U}$, and $d = (n_1 + m_1)^{\frac{2}{\epsilon}}$ *extra vertices* $q_1, q_2, \ldots, q_d$.

The edge set E. Add edge (u_i, v_j) if the element v_j in the set S_i. In addition, there is an edge from v_j to q_p $\forall 1 \leq j \leq n_1, 1 \leq p \leq d$. All edges have the weight of 1.

Vertex load and capacity. The load and capacity of the set vertex u_i are $L(u_i) = |S_i|$ and $C(u_i) = 1 + L(u_i)$. The load and capacity of the element vertex v_j are $L(v_j) = d - 1$ and $C(v_j) = d - 0.5$. All extra vertices have the load of 0 and capacity of $n_1 - 1 + \frac{n_1}{d}$.

Next, we prove that if $\mathcal{I}$ has a set cover of size k then there exist a seeding set $A \subset V$ such that $|F(A)| > d$. Otherwise, for all $A \subset V$ and $|A| \leq k$, $|F(A)| < n_1 + m_1$.

Assume that $\mathcal{I}$ has a set cover $\mathcal{SC}$ of size k, then we will select the set $A = \{u_i | S_i \in \mathcal{SC}\}$ as the seeding set. Initially, each vertex $u_i \in S$ redistributes 1 unit of load to each of its $|S_i|$ neighbors. Since each element is covered by at least one set in $\mathcal{SC}$, each vertex v_j receives the additional load of at least 1. At the next round, v_j has the load at least $L(v_j) \geq (d - 1) + 1)$, which is higher than its capacity, and is failed. When v_j fails, it equally redistributes $L(v_j)/d \geq 1$ load to its d extra neighbor vertices. The load of each extra vertex q_p after receiving load from all failed element vertices is $L(q_p) \geq n > C(q_p)$, hence all extra vertices are failed. The cascading process stops with $|F(A)| = d + n_1 + k$ failed vertices.

In the case $\mathcal{I}$ has no set cover of size k, we will show the optimal seeding set can cause at most $n_1 + k$ nodes fail in $\mathcal{I}'$. Let A be an arbitrary optimal seeding set. We observe that a set vertex only fails when it is selected in the seeding set since it has no incoming edge. Thus, there are at least $m_1 - k$ set vertices which are not in A. We can replace any extra vertex $q_p \in A$, if there are any, by a unselected set vertex without decreasing the number of failed nodes. Next, suppose that there exists an element vertex $v_j \in S$, we can also replace it by a set vertex. If v_j is adjacent to some vertex $u_i \in A$, we can remove v_j from A while maintaining the same number of failed nodes. If v_j is not adjacent to any vertex in A, we just replace v_j by one of its neighborhood

set vertex u_i. u_i will make v_j fail, so the number of failed nodes caused by A is not decreased. So, we can replace extra and element vertices in A such that A contains only set vertices. Since, there is no set cover of size k, there at least one vertex is not adjacent to any vertex in A ,i.e, the number of failed element vertices is at most $n_1 - 1$. Each failed element vertex v_j is adjacent to at most 2 set vertices, hence its load is at most $L(v_j) \leq d + 1$. Each extra vertex q_p receives at most $(d + 1)/d$ redistributed load from failed element vertices which are accumulated to at most:

$$\frac{(n_1 - 1)(d + 1)}{d} = n_1 - 1 + \frac{n_1 - 1}{d} < n_1 - 1 + \frac{n_1}{d} = C(q_p)$$

Thus, there is no extra vertex fails. The total failed nodes caused by A is at most $n_1 + k < n_1 + m_1$.

Now suppose that we have polynomial algorithm $\mathcal{A}$ which approximates CasCN problem within $n^{1-\epsilon}$, we can decide the set cover problem as follows. For any instance $\mathcal{I}$ of the Set Cover problem, we construct the instance $\mathcal{I}'$ as above in polynomial time as d is a polynomial function of n_1 and m_1. Now, if $\mathcal{I}$ has a set cover of size k, the optimal A_{opt} seeding set causes at least d vertices fail in $\mathcal{I}'$. The algorithm $\mathcal{A}$ approximate the optimal solution within $n^{1-\epsilon}$ ($n = n_1 + m_1 + d$, the number of vertices in $\mathcal{I}'$), so it finds a seeding set $\mathcal{A}(\mathcal{I}')$ whose causes at least $(m_1 + n_1)$ vertices fail:

$$\begin{aligned} |F(\mathcal{A}(\mathcal{I}')| \geq \tfrac{|F(A_{opt})|}{n^{1-\epsilon}} &> \tfrac{d}{(d+m+n)^{1-\epsilon}} \\ &> \tfrac{d}{(2d)^{1-\epsilon}} > \tfrac{d^{\epsilon}}{2} \\ &= \tfrac{(m_1+n_2)^2}{2} > m_1 + n_1 \end{aligned}$$

On the other hand, if $\mathcal{I}$ has no set cover of size k, then the optimal seeding set A_{opt} of $\mathcal{I}'$ causes less than $(m_1 + n_1)$ vertices fail. We have:

$$|F(\mathcal{A}(\mathcal{I}')| \geq |F(A_{opt})| < (m_1 + n_1)$$

It implies the $\mathcal{I}$ has a set cover of size k if and only if $|F(\mathcal{A}(\mathcal{I}')| > (m_1 + n_1)$. Hence, we can use $\mathcal{A}$ to decide the Set Cover problem in polynomial time i.e. $P = NP$. □

2.3 Cascading Potential and Derived Algorithms

In this section, we introduce a new metric to measure the node importance under the cascading failure, and then apply it to design efficient algorithms for CasCN. To evaluate the vulnerability of networks under the load redistribution, previous works in the literature propose various ranking methods and measure the effect of attacking top k nodes. However, these methods consider very limited topological information, hence may miss the most critical nodes. In [7, 36, 53], the authors solely use the load as the criterion to rank nodes. The failure of a high load node intuitively tends to cause a large number of nodes fail as it redistributes a large amount of large to its neighbors, but cascading failures started from a small load node at the right position may result in a larger number of failed nodes [54]. Wang et al. [54] overcome this shortcoming by directly assessing the effect of the cascading process, the number of failed nodes, which is triggered by the evaluated node. Nevertheless, the direct impact is one the top factors, they fail to incorporate the indirect impact into the node importance. When multiple nodes are attacked in the network, the indirect impact of a node is the base for the direct impact of other nodes. Next, we introduce a new metric which considers both direct and indirect impact of the node.

2.3.1 Cascading Potential

The cascading potential of a node is defined as combination of all possible impacts a node causes in the network under the cascading effect. Let's consider the failure of node u. For any other node v, there are two possible impacts that u can induce on v:

- *Failure impact*. The failure of u leads to the failure of v.
- *Load impact*. The failure of u makes the load of v increase but not enough to fail.

The overall *failure impact* and *load impact* of u in the network are defined as the number of failed nodes and the total of increased load of unfailed nodes, respectively. The *cascading potential* of u is the linear combination of these factors:

$$\mathcal{C}(u) = \frac{|F(\{u\})|}{n} + \frac{\sum_{v \in V - F(\{u\})} \Delta L_u(v)}{\sum_{v \in V - F(\{u\})} (C(v) - L(v))}$$

where $F(\{u\})$ is the set of failed nodes when u fails and $\Delta L_u(v)$ is the additional load that v receives due to the failure of u.

In this formula, we normalize both the failure and load impacts to avoid the unit difference. The failure impact is divided by the number of nodes, hence is at most 1 when all other nodes fail. Similarly, the load impact is divided by the total of capacity-load difference of unfailed nodes and achieves the maximum value 1 when all remained nodes are at the edge of failure, i.e., the most vulnerable state of the network.

The role of the load impact. In the formulation of the cascading potential, the load impact plays an important role to provide a better assessment of the network vulnerability comparing to the metric in [54]. If only one node is attacked, it is obviously to choose the node which maximizes the number of failed nodes, i.e., to use Wang et al.'s metric. However, when multiple nodes are attacked, we need to consider the co-impact of attacked nodes to trigger a large size cascading failure. The load impact is bridge connecting the impact of these nodes since the load impact of a node is the base for the failure impact of other nodes. For example, if u has the maximum load impact of 1 and the network is strongly connected, then attacking any node after u can take down the whole network. Thus, the cascading potential evaluate the importance of nodes more comprehensively.

2.3.2 Cascading Potential Algorithm

Intuitively, we can use cascading potential directly to design an algorithm for CasCN. We first compute the cascading potential of all nodes, then select top k as attacked nodes. The algorithm is described in Algorithm 1.

Algorithm 1 Cascading Potential Algorithm

Require: A network $G = (V, E)$, an integer k.
Ensure: A set S of k attacked nodes.
Compute the cascading potential of all nodes
Sort nodes in non-increasing order of the cascading potential $\mathcal{C}(u_1) \geq \mathcal{C}(u_2) \geq ... \geq \mathcal{C}(u_n)$
Initialize $S \leftarrow \emptyset$
$j \leftarrow 1$
for $i = 1$ to k **do**
    $S \leftarrow S \cup \{u_i\}$
end for
Return S

Time complexity. It takes at most $O(m)$ to compute the cascading potential of each node. Thus, the total running time is $O(nm + n \log n)$.

2.3.3 Adaptive Cascading Potential Algorithm

The Cascading Potential algorithm runs fast, but it neglects an important property of the cascading failure: the overlapped impact of selected nodes. Let consider two nodes u and v which both have failure impact on node z. If u is selected before v, then v has no impact z as z is already failed. As a consequence, some nodes have high impact initially will have small impact at the late of the selection process. We can improve the performance of the algorithm by updating the impact of remained nodes on the fly. More specifically, at the i^{th} iteration, the impact (failure or load impact) of node u on failed nodes (due to the selection of first $i - 1$ attacked nodes) will be subtracted from the initial impact of u. After that, the node with highest remained impact will be selected.

The crucial problem is how to update the impact of nodes efficiently. We may naively keep the list of impacted nodes for each node u. At each iteration, we compare the list of impacted nodes and the set of failed nodes to update the subtract the impact on failed nodes. This can result in $\Omega(n^3)$ running time for each iteration which is very time consuming. We reduce the updating time by reversing the process. Each node v will keep two lists of nodes: the list $FI[v]$ contains nodes which have failure impact on v and the list $LI(v)$ contains nodes which have load impact on v. Since the load impact

Algorithm 2 Adaptive Cascading Potential Algorithm

Require: A network $G = (V, E)$, an integer k
Ensure: A set S of k attacked nodes.

```
for each v ∈ V do
    Initialize FI[v] ← ∅, LI[v] ← ∅
end for
for each u ∈ V do
    Compute C(u)
    for each v ∈ F({u}) do
        FI[v] ← FI[v] ∪ {u}
    end for
    for each v: ΔL_u(v) > 0 and v ∉ F({u}) do
        LI[v][u] ← ΔL_u(v) / Σ_{z∈V−F({u})}(C(z)−L(z))
    end for
end for
Initialize S ← ∅
for each u ∈ V do
    Mark[u] ← False
end for
for i = 1 to k do
    u ← arg max_{v∈V\F+(S)}{C(v)}
    S ← S ∪ {u}
    for each v ∈ F(S) do
        if Mark[v] == False then
            Mark[v] ← True
            for each u ∈ FI[v] do
                C(u) ← C(u) − 1/|V|
            end for
            for each u ∈ LI[v] do
                C(u) ← C(u) − CL[v][u]
            end for
        end if
    end for
end for
Return S
```

of other nodes on v are different, we use $LI[v][u]$ to store the load impact of u on v after the normalization. When v is failed, the impact of nodes in its lists will be updated. The crucial point is that each node only fails once, thus the running time is reduced significantly. The algorithm with adaptive cascading potential is described in Algorithm 2.

Time complexity. Since each node has impact on at most n nodes, the total size of all FI and LI lists are at most n^2. The number of updates is bounded by the total size of FI and LI lists. Therefore the total running time is $O(nm + n^2 + kn)$.

2.3.4 Fully Adaptive Cascading Potential Algorithm

On the line of cascading potential based algorithms, we continue to improve the solution's quality by spending more time to calibrate the cascading potential of nodes. After a node u is attacked, the network state is changed with new failed nodes and load updates; and this may decrease (as discussed in the proceeding part) or increase the impact of a node. The failure of u adds load to many nodes and makes them more vulnerable. Although the impact of a remained node v is deducted by the impact on failed nodes, it can still increase since other nodes are easier to be failed. We can fully update the cascading potential of each node as follows. After selecting a new node, we simulate the cascading failure triggered by it and obtain a new graph of remained nodes. In this graph, the load of a node is the load when the cascading process stops. We then can evaluate the cascading potential of all nodes in the updated graph and select one with highest value. We present the algorithm in Algorithm 3.

Algorithm 3 Fully Adaptive Centrality

Require: A network $G = (V, E)$ and an integer k.
Ensure: A set S of k attacked nodes.
  Initialize $S \leftarrow \emptyset$
  for $i = 1$ to k **do**
    Compute the cascading potential of all nodes in G
    Select u as the node with highest cascading potential
    $S \leftarrow S \cup \{u\}$
    Update node loads and remove all failed nodes in G with the failure of u
  end for
  Return S

Time complexity. We need to compute the cascading potential of all nodes to select a new one with time $O(nm)$. Thus the total running time is $O(kmn)$. However, the

algorithm may run much faster than the worst case time since the size of the updated graph decreases when a new node is selected.

2.4 Cooperating Attack Algorithm

The key to connect the impact of multiple nodes in the above algorithms is the load impact which is a connection link when the network is robust. In this case where nodes are high failure tolerant, i.e., the gap between the capacity and load is big, the failure impact of each node is small. Thus, nodes with high load impacts tend to be selected.concentrate If the load of these nodes are scattered to many nodes, they are not linked together to make other nodes fail. As a consequence, there are a large number of nodes whose loads are increased, but there is only a few failed nodes. We incidentally try to maximize the total load impact instead of the failure impact – the objective function. We need a better strategy which builds a strong connection between selected nodes to increase the number of failed nodes. To fulfill this goal, the new strategy should satisfy following features:

- The redistributed load of selected nodes should be concentrated on certain nodes to fail them. If early selected nodes redistributed load to a set of nodes, then later selected nodes should also redistribute load to this set. It is said that selected nodes are cooperating in redistributing load to make more nodes fail.
- Selected nodes should cooperate to make high load nodes fail. The failure of high load nodes can expand the cascading failure further. However, if high load node preference reduces the number of failed nodes, the new strategy should avoid blindly favoring to fail high load nodes.

We design a new evaluation function, the efficiency, of nodes with properties that tailor the selection process to embrace both desired features. Firstly, we give higher evaluation to nodes which redistributes its load to load-increased nodes. If the failure of u pushes an additional load $\Delta L_u(v)$ on v, then the impact of u on v is defined by:

$$\gamma(u, v) = \frac{\Delta L_u(v)}{C(v) - L(v)}$$

when $\Delta L_u(v) + L(v) \leq C(v)$. Since it requires $C(v) - L(v)$ additional load to make v fail, we can interpret that u makes a fraction $\frac{\Delta L_u(v)}{C(v)-L(v)}$ of v fail.

The new impact function implies that if the more load a node receives, the more likely the new selected node will redistribute load on it. On the other hand, the evaluation of u is higher if the loads of its neighbors are increased. This implication is stated in Proposition 2.1.

Proposition 2.1. *For any node v at two points of time, if v receives more load at the second time point, i.e., $L_2(v) > L_1(v)$, then the impact of other node u with the same redistributed load ΔL is higher at the second time point: $\gamma_2(u, v) \geq \gamma_1(u, v)$.*

Proof. We have:

$$\gamma_2(u, v) = \frac{\Delta L}{C(u) - L_2(u)} > \frac{\Delta L}{C(u) - L_1(u)} = \gamma_1(u, v)$$

□

Note that we assume u redistribute the same load on v in the Proposition 2.1, i.e. the load of u is the same at two points of time. In fact, the load of u may increase due to the selection of previous nodes, thus the evaluation of u even increases more at the second point of time.

To fulfill the second feature, we assign higher values to high load nodes which are impacted. The value of a node with load L is:

$$\sigma(L) = \frac{e^L}{1 + e^L}$$

The function $\sigma(L)$ is monotone increasing and in the range $0.5 \leq \sigma(L) < 1$. The monotone increasing of the function shows the preference toward high load nodes. Recall that the main goal is to increase the number of failed nodes, so even nodes with the lowest load have the value at least half of the highest value nodes.

Next, we will define the efficiency of selecting u via the impact on v. Intuitively, u makes $\gamma(u, v)$ fraction of v fail and v has value of $\sigma(L(v))$, thus the efficiency of u

represented on v is:

$$\lambda(u, v) = \gamma(u, v)\sigma(L(v))$$

Finally, we obvious should take into account the number of failed nodes when evaluating node u. The overall efficiency of u is the total of the number of failed and the efficiency on unfailed nodes:

$$\lambda(u) = |F(\{u\})| + \sum_{v \in V \setminus F(\{u\})} \lambda(u, v)$$

The efficiency evaluation shows several notable properties which servers our design goal as followings:

Increase the number of failed nodes first. If u makes z fail and has efficiency $\lambda(u, v)$ on the unfailed node v, then the contribution of z to the overall efficiency of u is always higher than v since $1 \geq \gamma(u, v)\sigma(L(v))$.

Avoid redistributing load to impossible-to-fail nodes. If node v needs too much additional load before failing, it will be ignored in efficiency evaluation of nodes as stated in the Proposition 2.2.

Proposition 2.2. *Given two nodes u and v with fixed load $L(v)$, the efficiency of u on v is monotone decreasing and goes to 0 when the capacity $C(v)$ of v increases and goes to infinity.*

Proof. It is easy to see that $\gamma(u, v)$ is monotone decreasing and goes to 0 when $C(v)$ increases and goes to infinity. In addition, $\sigma(L(u))$ is a constant, so the efficiency $\lambda(u, v) = \gamma(u, v)\sigma(L(v))$ decreases and goes to 0. □

Not favoring high load nodes with all cost. We consider the case the capacity is linear to the load, a common setting in the reality to guarantee the safety of nodes. In this case, even the load of node v is extremely large, it is still ignored as shown in Preposition 2.3.

Proposition 2.3. *Suppose that the capacity $C(v)$ is linear to the load $C(v) = T * L(v)$ with constant factor T. Then, the efficiency of any node u on v goes to 0 when the load $L(v)$ goes to infinity.*

Proof. We have:

$$\begin{aligned}\gamma(u,v)\sigma(L(u)) &= \frac{\Delta L_u(v)}{C(v)-L(v)}\frac{e^{L(v)}}{1+e^{L(v)}} \\ &< \frac{\Delta L_u(v)}{(T-1)L(v)}\end{aligned}$$

The function goes to 0 when $L(u)$ goes to infinity. □

Based on the efficiency evaluation, we propose the Cooperating Attack (CA) algorithm with the same manner of Fully Adaptive Cascading Potential algorithm. The algorithm also selects nodes one by one. After updating the state of the network, the node with highest efficiency is selected. The whole algorithm is described in Algorithm 4.

Algorithm 4 Cooperating Attack (CA) Algorithm

Require: A network $G = (V, E)$ and an integer k.
Ensure: A set S of k seed nodes.
  Initialize $S \leftarrow \emptyset$
  for $i = 1$ to k **do**
    Evaluate the efficiency of all nodes in G
    Select u as the node with the highest efficiency
    $S \leftarrow S \cup \{u\}$
    Update node loads and remove all failed nodes G with the failure of u
  end for
  Return S

Time complexity. Similarly to the Fully Adaptive Cascading Potential algorithm, the total running time is $O(kmn)$.

2.5 Experimental Evaluation

In this section, we demonstrate the experimental results on both synthesized and real power networks. We first test the performance of the proposed algorithms in the comparison with current attacking strategies in the current literature [53, 54]. These

strategies sort nodes based on some criterion and select top k nodes as attacked nodes. The sorting criteria are:

- Largest load (HL).
- Lowest load (LL).
- Highest percentage of failure (PoF). The percentage of failure of a node u is the fraction of nodes is failed when u fails.
- Highest risk if failure (RIF). RIF of a node u is the ration between its load and the total load of its neighbor nodes.

We omit the required redundancy of Wang et al. [54] since it provides the same order of nodes as RIF. After that, we evaluate the robustness of networks under different failure tolerance schemes to identify the suitable network design considering the effect of cascading failures.

2.5.1 Datasets

Real network. We use the Western North American (WNA) power grid network [55] with 4941 substations and 6594 transmission lines to run experiments. However, the dataset is lacking of load and capacity information of nodes, thus we use the method in [56] to assign the load and capacity for each node. The initial load of of node u is given by $L(u) = d(u)^{\beta}$ where $d(u)$ is the degree of u and α is a tunable parameter. This assignment method is reasonable as the load of the node is shown to be scaled with its degree [59]. Vertex capacities are assigned based on three different schemes.

Normal networks. In normal networks, the capacity $C(u)$ of each node u is proportional to its initial load $L(u)$:

$$C(u) = T * L(u)$$

where T is a constant representing the *system tolerance*. The higher T is, the more endurance the network is under the cascading failure.

Safe networks. In safe networks, the node capacities are assigned in two phases. First, the capacity $C(u)$ of each node u is scaled as the normal network, i.e:

$$C(u) = T * L(u)$$

Then, then capacities of all nodes are raised to satisfy the $N-1$ failure tolerance criterion in which the failure of any node will cause no additional failed nodes. It means that any node u will not fail when it receives the redistributed load from any of its neighbor. The capacity of u will be:

$$C(u) = \max\{C(u), \max_{v \in N^-(u)} \{L(u) + w(v,u)L(v)\}\}$$

Scaled Safe networks. In contrast to safe networks, scaled safe networks are formed by raising the node capacities to satisfy $N-1$ failure tolerance criterion first, then be scaled up later. In particular, the network is made safe by assigning the capacity of each node u as:

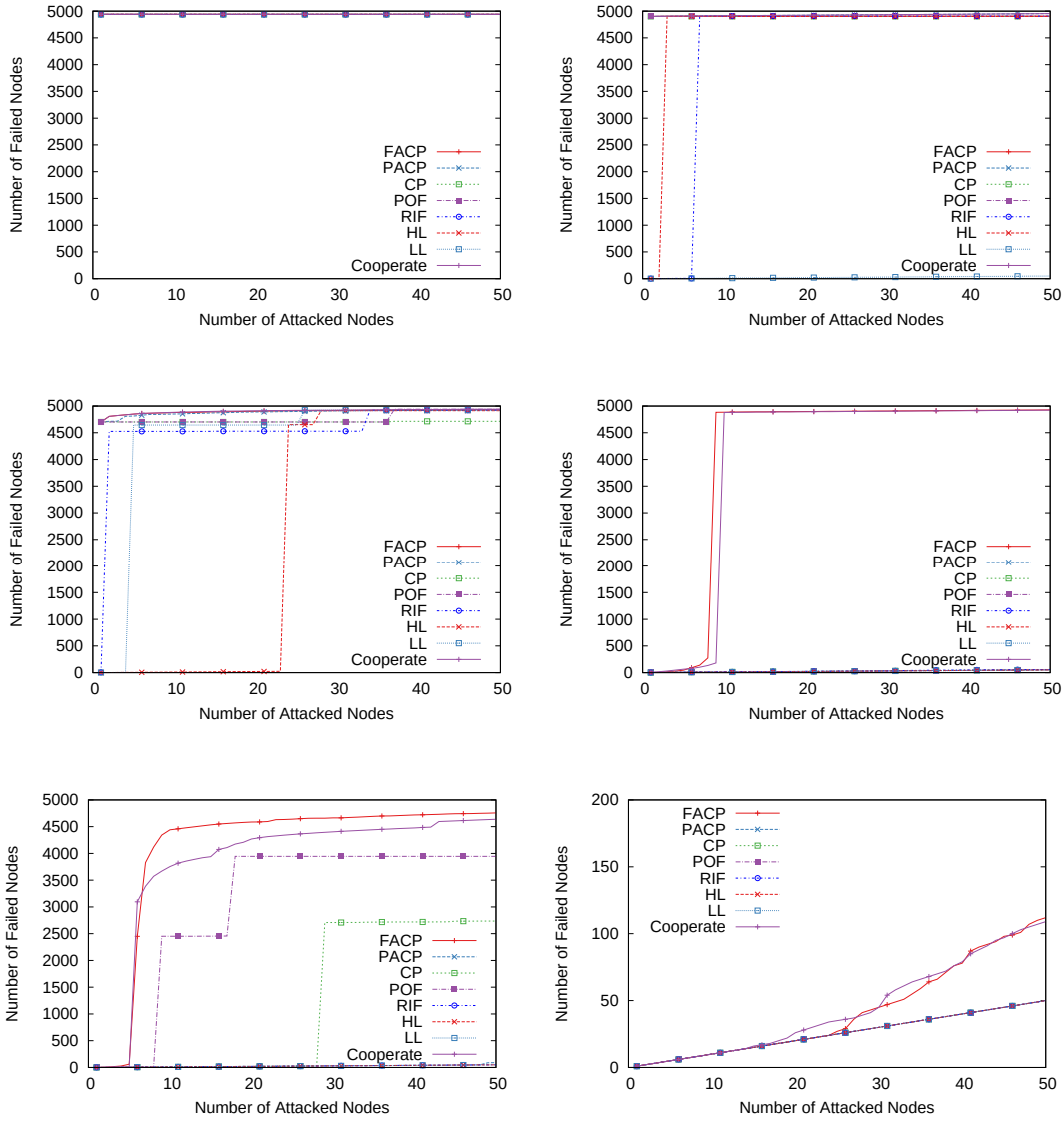
$$C(u) = \max_{v \in N^-(u)} \{L(u) + w(v,u)L(v)\}$$

Then scale up the capacity of u to $C(u) = T * C(u)$.

Synthesized Networks. We also run the experiments on synthesized networks generated by Erdos-Renyi random network model [26]. Each network has 5000 vertices with the average degree of 4 or 8. The other parameters of the network is generated similar to above schemes.

2.5.2 The performance of Different Algorithms

We first compare the performance of proposed algorithms and the previous works. We run experiments on networks with different system tolerance values and $\beta = 1$. The results are shown in Fig. 2-3, 2-4, and 2-5. When networks have small tolerance values, they are so vulnerable under any attacking strategy. The failure of one or two nodes can lead to the failure of almost the whole network. On the other hand, when T is high, the network can endure multiple attacks without failing. These figures also show



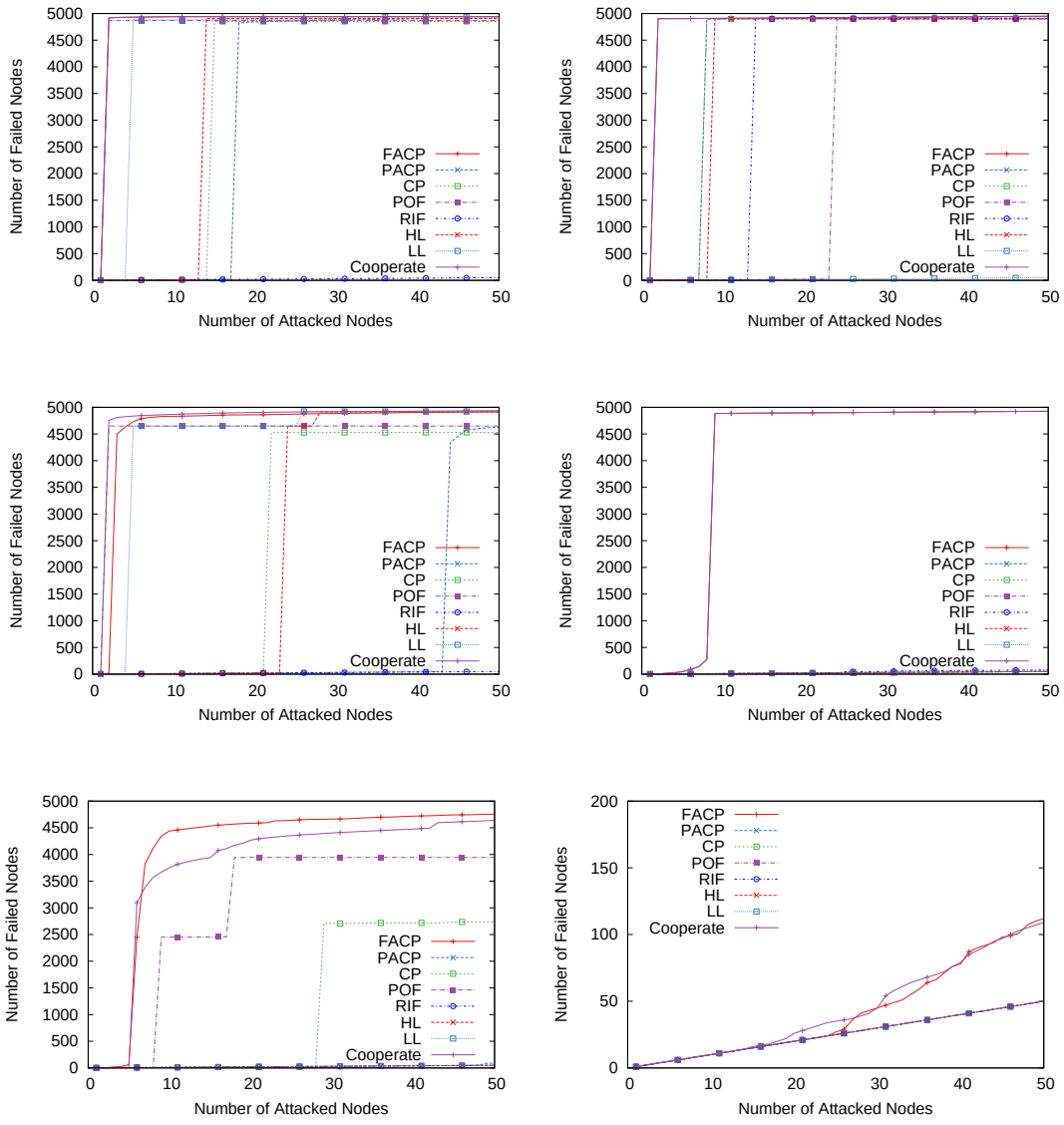
A WSN network

B Synthesized network ($d = 4$)

Figure 2-3. Vulnerability of WSN network under normal setting

that previous attacking strategies do not work well on safe networks. On safe networks, the performance of FACP and CA algorithms are the best since they can adapt to the change of network status to choose nodes for attacking. Additionally, when the network is vulnerable, FACP shows better performance. Early attacked nodes push remaining nodes to the boundary of failure. The redistributed load of later attacked nodes can make them fail easily. The situation is changed when networks are robust. CA algorithm

optimizes the number of failed nodes at each iteration, so its produced seed set make more nodes fail.



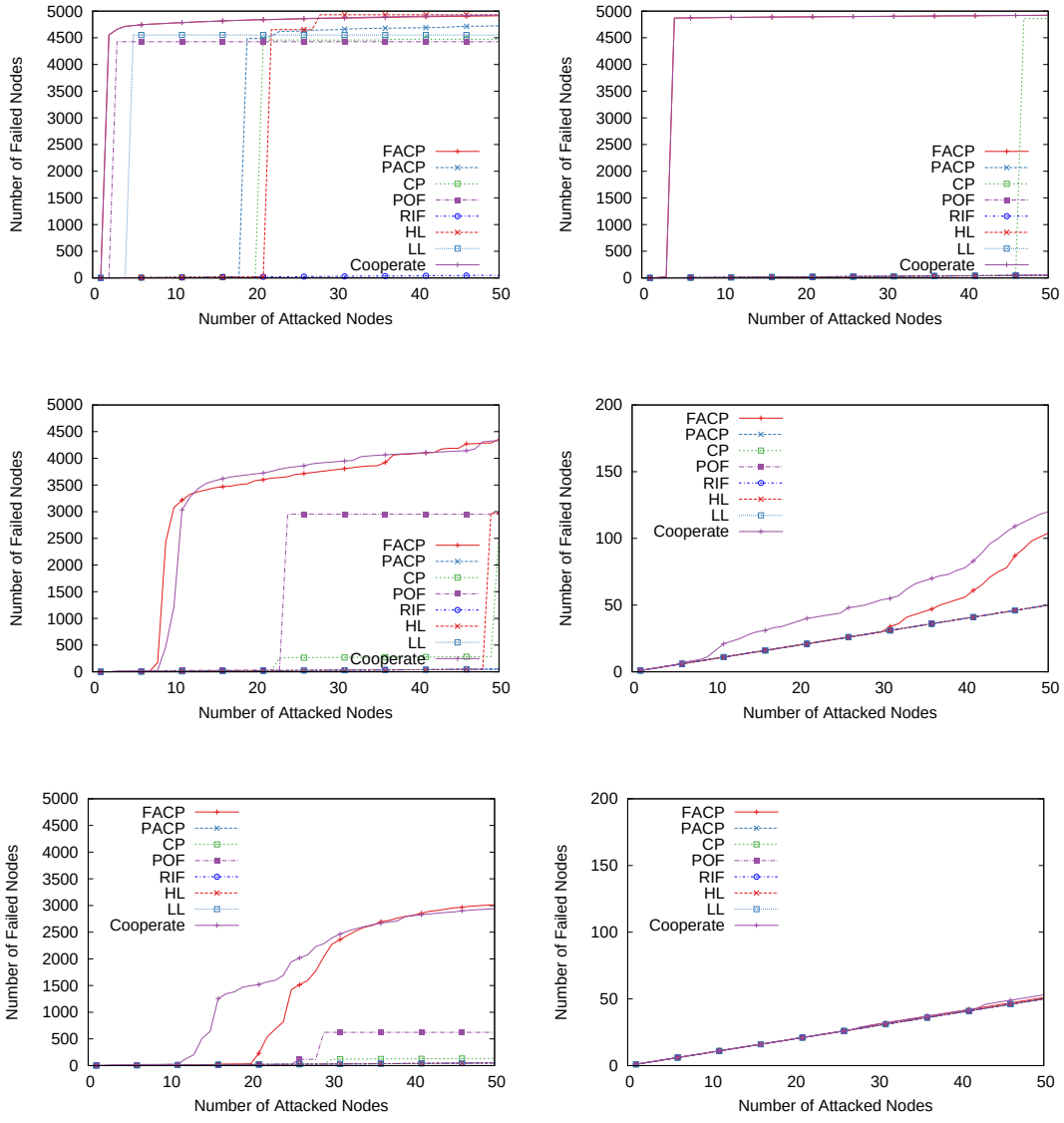
A WSN network

B Synthesized network ($d = 4$)

Figure 2-4. Vulnerability of WSN network under safety setting

2.5.3 Network Robustness Under Different Settings

We observed that scaled safe networks are more robustness than safe networks and safe networks are stronger than normal networks. However, the first kind of networks often has highest total of capacities while it has the same node load as the



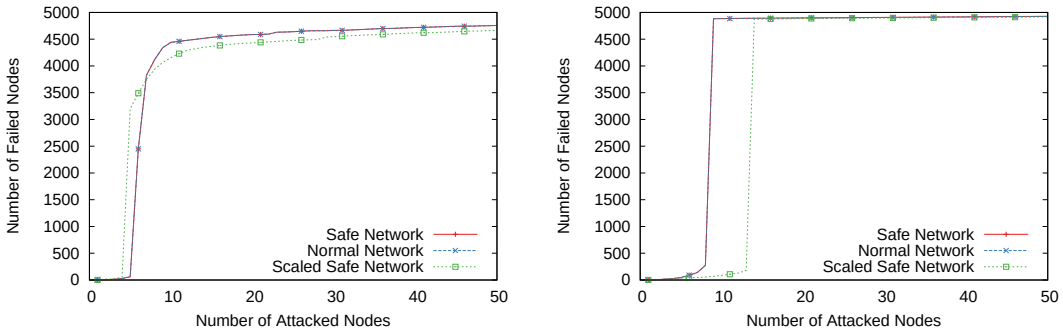
A WSN network

B Synthesized network ($d = 4$)

Figure 2-5. Vulnerability of WSN network under scaled safety setting

remained two networks. Thus, we set up the experiment to measure the robustness of each kind of networks as follows. We first generate the safe network, then we choose a suitable T value to generate normal and scaled safe network such that the total capacity is the same. Fig. 2-6 shows that scaled safe network are the most robust one. Normal and safe networks have very close robustness although the safe factor can help to avoid the first attack. When multiple nodes are attacked, N-1 failure tolerance setting does

not help much. The failure of the first node makes some other nodes reach the failure limit. Even a tiny additional load can make them fail and trigger a large cascading failure. Therefore, we observe that the N-1 safe criterion does not protect power networks under cascading attack.

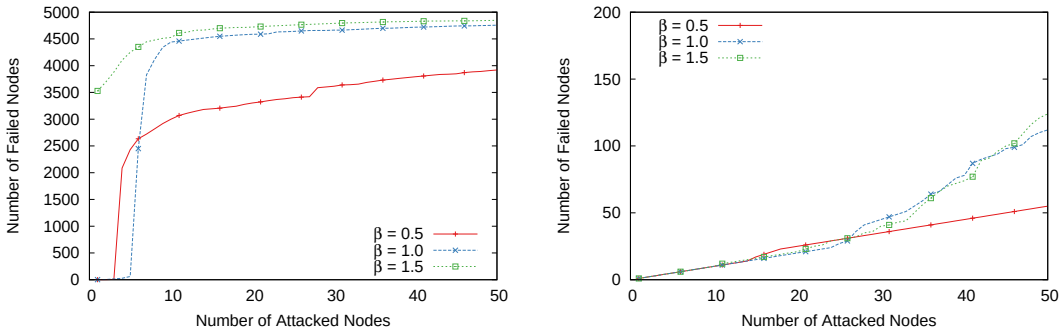


A WSN ($T = 2, \beta = 1$) B S4 ($T = 1.6, \beta = 1$)

Figure 2-6. Network robustness with different failure tolerance schemes

2.5.4 Vertex Load and Network Robustness

Now we vary the value of β and measure the robustness of networks under the FACP attacking strategy. As illustrated in Fig. 2-7, the higher β is, the more vulnerable the network is as the load is concentrated at a few nodes. These nodes become the Achilles' heel of the network. It suggests that we should distribute the workload of nodes in networks to make them less vulnerable.



A WSN ($T = 2, \beta = 1$) B S4 ($T = 1.6, \beta = 1$)

Figure 2-7. Network robustness under different load distribution

2.5.5 Network Topology and Network Robustness

We next evaluate the robustness of networks with different average degree. As shown in Fig. 2-8, the network with higher average degree is more robust than network with lower average degree. In the denser network, the load of a failed node is redistributed to a larger number of neighbors. Each neighborhood node receives a small fraction of the load, thus it can bear the additional load without failing.

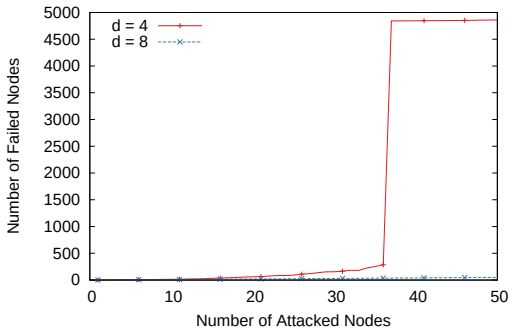


Figure 2-8. Network topology and robustness

2.6 Related Works

The cascading failure has attracted a lot of attention and been studied in various perspective [6, 22, 30, 40, 46, 53, 54, 60]. The structural vulnerability of power networks was studied in [6]. The authors showed that removing small fraction of highest degree nodes significantly reduces the connectivity of the network. After that, Hines et al. [30] studied the network vulnerability of different classes of scale-free networks including Erdos-Renyi, preferential-attachment, and small-world networks. They showed that different types of networks behave differently under node failures. There various models of cascading failures were later proposed to vulnerability of networks under targeted attack [7, 36, 53, 54]. However, these works mainly present different ranking methods for nodes and select most critical nodes. These methods fail to address the effect of the cascading process, hence miss to the most critical nodes.

2.7 Summary

In this chapter, we studied the critical node detection problem under load redistribution cascading model. Based on the new centrality which considers the cascading effect when evaluate the importance of nodes in networks, we develop various algorithms to identify critical nodes: one is purely based on the new centrality, one is based on partially adaptive centrality, and one is based on fully adaptive centrality. Among them, the fully adaptive centrality algorithm continuously updates the centrality of nodes and select the best one, hence achieves the highest performance among the three. However, this algorithm suffers when the network is highly tolerant to the cascading failure. We propose the cooperating attack algorithm which cooperates selected nodes to take down protected nodes with high capacity. The performance guarantee of the cooperating attack algorithm is supported by both theoretical and experimental results.

In addition, we use proposed algorithms to study the vulnerability of different safety settings. We find that networks with low density is extremely vulnerable under the cascading failure. In this kind of networks, the load of a failed node is redistributed to a small number of neighbors and can fails them easily. On the other hand, the load is shred to smaller portions in networks with high density. We also discover that even with network of the same topology and total capacity and load, the network safety depends a lot on the distribution of protection cost (the gap between the capacity and the load). These shred the light on designing safe networks.

CHAPTER 3
CASCADING FAILURE OF NODES IN INTERDEPENDENT NETWORKS

In the development of technology, infrastructure networks are more and more interdependent on each other to operate properly. To optimize the operation performance and reduce the economic spend, these networks tend to utilize the support from other ones. A typical example is the Smart Grid in which the power network uses the communication network to exchange operational information and the communication network uses the electricity from the power network to operate. In the meanwhile, such growing interdependencies also dramatically impact the vulnerability of these networks since a network is not only exposes to threats to themselves but also to the cascading failures induced by from other networks. In a typical attacking point of view, an attacker would first exploit the network weaknesses, and then only needs to target on some critical nodes in either power networks or their interdependent communication networks, whose corruptions bring the whole network down to its knees. In other words, nodes from power networks depend heavily on nodes from their interdependent networks and vice versa. Consequently, when nodes from one network fail, they cause nodes in the other network to fail, too. For instance, an adversarial attack to any essential Internet hosts, e.g. tier-1 ISPs such as Qwest, AT&T or Sprint servers, once successful, may cause tremendous breakdowns to both millions of online services and the further large-area blackout because of the cascading failures. A real-world example is the wide-range blackout that affected the majority of Italy on 28 September 2003 [47], which resulted from the cascading failures induced by the dependence between power networks and communication networks. Therefore, in order to guarantee the robustness of power networks without reducing their performance by decoupling them from information systems, it is important to identify those critical nodes in interdependent power networks, beforehand.

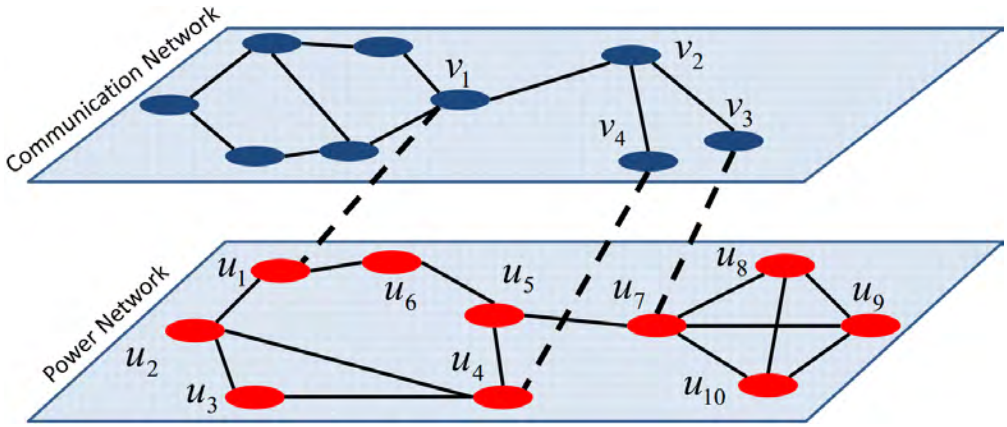


Figure 3-1. Example of Interdependent Power Network and Communication Network

There have been many studies assessing the network vulnerability [6, 8, 15, 27, 32, 45, 48]. Yet, these approaches are either designed only for single networks or heavily dependent on configuration models of interdependent networks. The existing approaches [5, 6, 8, 39] for single networks are based on various metrics, such as the degree of suspected nodes or edges [6], the average shortest path length [5], the global clustering coefficients [39], and the pairwise conductivity [6, 8] and so on. However, when applying into interdependent networks, their performances drop tremendously since these metrics fail to cast the cascading failures in interdependent networks. Later on, other researchers [15, 27, 32, 45, 48] studied the vulnerability assessment on interdependent networks, based on the size of largest connected component in power networks after cascading failures. Although they showed the effectiveness of this new metric, most of them focus on the artificial models of interdependent networks, i.e., random interdependency between networks, and ignore the detection of top critical nodes in real networks.

Let us consider a simple example of interdependent networks in Fig. 3-1, which illustrates a small portion of power network (lower nodes), communication network (upper nodes) and their interdependencies (dotted links). When we only take the single power network into account, the failure of u_7 destructs the power network more than that of u_1 since the largest connected component is of size 6 ($\{u_1, u_2, \ldots, u_6\}$) when u_7

fails, which is smaller than 9 ($\{u_2, \ldots, u_{10}\}$) when u_1 fails. However, if considering its interdependence upon the communication network, the failure of u_1 will destroy the power network more than that of u_7. This is because the failure of u_1 causes the failure of v_1 in the communication network, which further fails v_2, v_3, and v_4 since they are disconnected from the largest connected component. Due to their interdependence of the nodes v_4 and u_7 in the power network, these cascading failures finally result in the largest connected component of the power network to be $\{u_8, u_9, u_{10}\}$ of size only 3. Yet, the largest connected component remains the same as in a single power network after the failure of u_7. This example illustrates an important point that the role of one node could be totally different between single and interdependent networks with respect to the vulnerability assessment.

In this chapter, we investigate the vulnerability of interdependent networks when the cascading failures happen based on the connectivity of nodes. When studying interdependent networks, especially the power network and the communication network, it is well known to assume that a node will failure when it is disconnected from the largest connected component. Although the assumption does not capture the reality, it contains a realistic meaning. In real world, when a node is disconnected the largest connected component, it is almost removed from the source of power or information. In this chapter, we study the problem in the context of power network and communication network, but the results is applied for all systems that share the same cascading mechanism.

The rest of the chapter is organized as follows. In Section 3.1, we introduce the interdependent network model and the problem definition. After that, Section 3.2 includes the hardness and inapproximability results. The greedy framework is proposed in Section 3.3, along with the centrality metric. The experimental evaluation is illustrated in Section 3.4. The related work is presented in Section 3.6. Finally, Section 3.7 provides some concluding remarks.

3.1 Network Model and Problem Definition

In this section, we first introduce our interdependent network model and the well-accepted model of cascading failure. Using these models, we study the *Interdependent Power Network Disruptor* problem, to minimize the size of largest connected components in the power network after cascading failures by selecting a certain number of target nodes.

3.1.1 Interdependent Network Model

Considering an interdependent system, we abstract it into two graphs, $G_s = (V_s, E_s)$ and $G_c = (V_c, E_c)$, and their interdependencies, E_{sc}. G_s and G_c represent the power network and communication network respectively. Each of them has a set of nodes V_s, V_c and a set of links E_s, E_c, which are referred to as *intra-links*. In addition, E_{sc} are *inter-links* coupling G_s and G_c, i.e., $E_{sc} = \{(u, v)|u \in V_s, v \in V_c\}$. A node $u \in V_s$ is functional if it is connected to the giant connected component of G_s and at least one of its interdependent nodes in G_c is in a working state. The whole interdependent system is referred to as $\mathfrak{I}(G_s, G_c, E_{sc})$.

3.1.2 Cascading Failures Model

We use a well-accepted cascading failure model, which has been validated and applied in many previous works [15, 27, 32, 45, 48]. Initially, there are a few critical nodes failed in network G_s, which disconnects a set of nodes from the largest connected component of G_s. Due to the interdependency of two networks, all nodes in G_c only connecting to failed nodes in G_s are also impacted, and therefore stop working. Furthermore, the failures cascade to nodes which are disconnected from the largest connected component in G_c and cause further failures back to G_s. The process continues back and forth between two networks until there is no more failure nodes.

3.1.3 Problem Definition

Definition 3 (IPND problem)**.** *Given an integer k and an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, which consists of two networks $G_s = (V_s, E_s)$, $G_c = (V_c, E_c)$ along with*

their interdependencies E_{sc}. Let $LG_s(T)$ be the size of the largest connected component of G_s after the cascading failures caused by the initial removal of the set of nodes $T \subseteq V_s$ in G_s. The IPND problem asks for a set T of size at most k such that $LG_s(T)$ is minimized.

In the rest of the chapter, the pairs of terms interdependent, networks and coupled networks, node and vertex, as well as edge and link, are used interchangeably.

3.2 Computational Complexity

In this section, we first show the NP-completeness of IPND problem by reducing it from maximum independent set problem, which further implies that IPND problem is NP-hard to be approximated within the factor $2 - \varepsilon$ for any $\varepsilon > 0$.

Theorem 3.1. *IPND problem is NP-complete.*

Proof. Consider the decision of IPND that asks whether the graph $G_s = (V_s, E_s)$ in an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$ contains a set of vertices $T \subset V_s$ of size k such that the largest connected component in $G_s[V_s \setminus T]$ after cascading failures is at most z for a given positive integer z. Given $T \in V_s$, we can compute in polynomial time the size of the largest connected component in G_s after the cascade failures when removing T by iteratively identifying the largest connected component and removing disconnected vertices in G_s and G_c. This implies IPND $\in$ NP.

To prove that IPND is NP-hard, we reduce it from the decision version of the maximum independent set (MIS) problem, which asks for a subset $I \subseteq V$ with the maximum size such that no two vertices in I are adjacent. Let an undirected graph $G = (V, E)$ where $|V| = n$ and a positive integer $k \leq |V|$ be any instance of MIS. Now we construct the interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$ as follows. We select $G_s = G$ and G_c to be a clique of size $|V_s|$. Since G_s and G_c have the same size in our construction, to construct the interdependency between G_s and G_c, we randomly match each node in V_s to some arbitrary nodes in V_c so as to form a one-to-one correspondence between V_s and V_c. An example is illustrated in Fig. 3-2. We show that there is an MIS of size

at most k in G iff G_s in $\mathfrak{I}(G_s, G_c, E_{sc})$ has an IPND of size $n - k$ such that the largest connected component in G_s after cascading failures is of size at most 1.

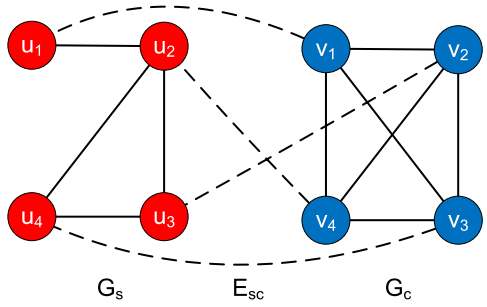


Figure 3-2. An example of reduction from MIS to IPND

First, suppose $I \subseteq V$ is an MIS for G with $|I| \leq k$. By our construction, the largest connected component of $G_s[I]$ has the size 1 since there is no more cascading failure in the clique G_c. That is, $V_s \setminus I$ is also an IPND of $\mathfrak{I}(G_s, G_c, E_{sc})$.

Conversely, suppose that $T' \subseteq V_s$ with $|T'| = n - k$ is an IPND of $\mathfrak{I}(G_s, G_c, E_{sc})$, that is, the largest connected component of $G_s[V_s \setminus T']$ is of size 1. We show that $V_s \setminus T'$ is also an MIS of G. Since the failure of nodes in G_c will not cause any cascading failure in G_s, the largest connected component of $G_s[V_s \setminus T']$ is of size 1 iff $V_s \setminus T'$ is an independent set in G_s. That is, $V_s \setminus T'$ is also an MIS of G. □

As IPND is NP-complete, one will question how tightly we can approximate the solution, leading to the theory of inapproximability. The inapproximability factor gives us the lower bound of near-optimal solution with theoretical performance guarantee. That said, no-one can design an approximation algorithm with a better ratio than the inapproximability factor. Then, we show that the above reduction implies the $(2 - \varepsilon)$-inapproximability factor for IPND in the following corollary.

Corollary 1. *IPND problem is NP-hard to be approximated into* $2 - \varepsilon$ *for any* $\varepsilon > 0$.

Proof. We use the reduction in the proof of Theorem 3.1. Suppose that there is a $(2 - \varepsilon)$-approximation algorithm $\mathcal{A}$ for IPND. Then $\mathcal{A}$ can return the largest connected component in G_s of size less than 2 in our constructed instance if the optimal size is 1.

Thus algorithm $\mathcal{A}$ can be applied to solve MIS on G in polynomial time because this size is integral. This contradicts to the NP-hardness of MIS. □

3.3 Greedy Framework for IPND Problem

In this part, we present different algorithms to detect the top critical nodes using the greedy framework, which has been illustrated as one of the most popular and effective approaches to solve hard problems. The idea is to iteratively choose the most critical node, whose removal degrades the functionality of the network as much as possible. In detail, we propose three following different strategies to select a critical node in the system at each iteration:

1) Select a node that maximizes the number of failed nodes after the cascading failure.

2) Select a node that decreases the structural strength of the system as much as possible. That is, when the number of removed nodes is large enough, the system will become weak. Therefore, the number of failed nodes increases considerably under the effect of cascading failures.

3) Select a node that not only increases the number of failed nodes but decreases the structural strength as well.

In the rest of this section, we describe three algorithms corresponding to the above strategies.

3.3.1 Maximum Cascading (Max-Cas) Algorithm

In maximum cascading (Max-Cas) algorithm, we iteratively select a node u that leads to the most number of new failed nodes, i.e., the maximum marginal gain to the current set of attacked nodes T. When a new node u fails, it results in a chain of cascading failures. The number of new failed nodes, referred to as *cascading impact number*, can be computed by simulating the cascading failures with the initial set $T \cup \{u\}$ on the interdependent system $\mathfrak{I}$ as described in Section 3.1.2. However, the simulation of cascading failures is time-consuming due to its calculation of cascading failures

between two networks. Each step in the cascading requires to identify the largest connected component of each network.

To this end, we further improve the running time of our algorithm by reducing the number of simulations. The idea is only to check potential nodes whose removal creates at least one more failed node in the same network due to the cascading failures. That is, this node (or its coupled node) disconnects the network which it belongs to, i.e., it (its coupled node) is an articulation node of G_s (or G_c), which is defined as any vertex whose removal increases the number of connected components in G_s (or G_c). The reason is illustrated in the following lemma.

Lemma 1. *Given an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, removing a node $u \in V_s$ from the system causes at least one more node fail due to the cascading failure iff u (or its coupled node $v \in V_c$) is an articulation node in G_s (or G_c).*

Proof. If u is an articulation node of G_s, the removal of u will increase the number of connected components in G_s at least to two. By the definition of cascading failures in G_s, all nodes disconnected from the largest connected component will be failed. Similarly, when v is an articulation node of G_c, removing u causes v fail, then there is at least one more nodes in G_c fail. After that, these nodes make coupled nodes in G_s fail as well. On the other hand, if neither u nor v are articulation nodes, the removal of u only makes v fail, and the rest of two networks are still connected, which terminates the cascading failures. □

According to this property, the proposed algorithm first identifies all articulation nodes in both residual networks using Hopcroft and Tarjan's algorithm [31]. Note that this algorithm runs in linear time on undirected graphs, which is faster than one simulation of cascading failures. Thus, the running time of each iteration is significantly improved especially when the number of articulation nodes is small. Denote $Max - Cas(G_s, T, \{u\})$ as the impact number of u, Algorithm 5 describes the details to detect critical nodes. In Algorithm 5, since it takes $O(n)$ time to compute the cascading impact

number for each node and at most $|A| < n$ articulation nodes will be evaluated, the running time is $O(kn^2)$ in the worst case. In practice, the actual running time is much less due to the small size of A, which is further illustrate in Section 3.4.

Algorithm 5 Max-Cas Greedy Algorithm

Input: Interdependent system $\Im(G_s, G_c, E_{sc})$, an integer k
Output: Set of k critical nodes in $T \in V_s$
$T \leftarrow \emptyset$
for $i = 1$ to k **do**
  $A_s, A_c \leftarrow$ set of articulation nodes of G_s and G_c respectively
  $A \leftarrow \{u \in V_s | u \in A_s \vee ((u, v) \in E_{sc} \wedge v \in A_c)\}$
  if $A \neq \emptyset$ **then**
    $u \leftarrow \text{argmax}_{u \in A}$ Max-Cas$(G_s, T, \{u\})$
    $T \leftarrow T \cup \{u\}$
  else
    $u \leftarrow$ any node in $V_s \setminus T$
  end if
  Update $\Im[V_s \setminus T]$
end for
Return T

3.3.2 Iterative Interdependent Centrality (IIC) Algorithm

As one can see, Max-Cas algorithm prefers to choose nodes that can decrease the size of networks immediately. This can mislead the algorithm to select boundary nodes and affect its efficiency for large k since the residual networks still remain highly connected even many critical nodes have been removed. An alternative strategy is to identify the hub nodes which plays a role to connect other nodes together in the network. Actually, this strategy has been proved to be efficient in single complex networks by Albert *et al.* [6], in which the removal of a small fraction of nodes with highest degree centrality has been shown to fragmentize the network to small components. However, since this centrality method is only for single networks, the development of a new centrality measure is in an urgent need for interdependent systems.

Intuitively, this new centrality measure is required to capture both the intra-centrality (the centrality of nodes in each networks) and inter-centrality (the centrality formed by the interconnections between two networks). Given an interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, node u in V_s is more likely to be critical if its coupled node $v \in G_c$ is critical. Furthermore, when node u is considered as a critical node, its neighbors are also more likely to become important since the failures of these nodes can cause u fail. That said, the criticality of these nodes imply the criticality of their coupled nodes. To capture this complicated relation in interdependent systems, we develop an iterative method to compute the centrality of nodes, called *Iterative Interdependent Centrality* (IIC). Initially, the centralities of all nodes in G_s are computed by the traditional centrality, e.g., degree centrality, betweenness centrality, etc. After that, these centralities of nodes in G_s are reflected to coupled nodes in G_c and the centralities of nodes in G_c are updated based on the reflected values. The centralities of nodes in G_c continue to be reflected on nodes of G_s and update centralities of these nodes. Two key points of IIC are the *updating function* and the *convergence*.

3.3.2.1 Updating function

Considering the reflected values from the other network as the weight of nodes, we modify the *weighted degree* as the updated centrality of nodes, which is defined as

$$\mathcal{C}(u) = \alpha w(u) + (1 - \alpha) \sum_{v:(u,v)\in E} \frac{w(v)}{d_v}$$

where $w(\cdot)$ is the reflected values (or the weight of nodes) and the reservation factor α lying in the interval $[0, 1]$. The underlying reason we use weighted degree is that a node is usually more critical if most of its neighbors are critical nodes. The reservation factor shows that the importance of each node is not only dependent on the reflected values from the other network, but also the role in its own network.

3.3.2.2 Convergence

Next, we show that the centralities of nodes can be computed based on matrix multiplications and prove the convergence via this property. Let $\mathbf{x}^t = [x^t_{v^s_1}, x^t_{v^s_2}, \dots, x^t_{v^s_n}]$, $\mathbf{y}^t = [y^t_{v^c_1}, y^t_{v^c_2}, \dots, y^t_{v^c_n}]$ be the normalized centrality vector after t^{th} iteration of G_s and G_c. Suppose that two interdependent nodes have the same position vectors $\mathbf{x}^t$ and $\mathbf{y}^t$, i.e., v^s_i and v^c_i are interdependent. Then, we have

$$x^t_u = \alpha y^{t-1}_u + (1-\alpha) \sum_{v:(u,v)\in E_s} \frac{y^{t-1}_v}{d_v}, \quad \forall u \in V_s$$

$$y^t_u = \alpha x^{t-1}_u + (1-\alpha) \sum_{v:(u,v)\in E_c} \frac{x^{t-1}_v}{d_v}, \quad \forall u \in V_c$$

Note that if we divide these vectors by a constant, then they still represent the centrality of nodes in the systems. Thus, after each iteration, these centrality vectors are divided by some constants C_s and C_c which are chosen later to prove the convergence.

$$\mathbf{x}^t = \mathbf{x}^t / C_s, \ \mathbf{y}^t = \mathbf{y}^t / C_c$$

Let M^s and M^c be $n \times n$ matrices such that

$$M^s_{uv} = \begin{cases} \alpha & \text{if } u = v \\ d_v^{-1} & \text{if } (u,v) \in E_s \\ 0 & \text{otherwise} \end{cases}$$

$$M^c_{uv} = \begin{cases} \alpha & \text{if } u = v \\ d_v^{-1} & \text{if } (u,v) \in E_c \\ 0 & \text{otherwise} \end{cases}$$

Then the relationship between $\mathbf{x}^t$ and $\mathbf{y}^t$ is rewritten as:

$$\mathbf{x}^t = \frac{M^s \mathbf{y}^{t-1}}{C_s}, \ \mathbf{y}^t = \frac{M^c \mathbf{x}^{t-1}}{C_c}$$

Therefore

$$\mathbf{x}^t = \frac{M^s M^c \mathbf{x}^{t-2}}{C_s C_c} = \frac{M \mathbf{x}^{t-2}}{C_s C_c}$$

where $M = M^s M^c$ is called the characteristic matrix. Next, we analyze the condition of this matrix to guarantee that $\mathbf{x}^t$ converges by using the Jordan canonical form of M, defined as follows.

Theorem 3.2 (Jordan Canonical Form [50]). *Any $n \times n$ matrix M with n eigenvalues $|\lambda_1| \geq |\lambda_2| \geq ... \geq |\lambda_n|$ can be represented as $M = PJP^{-1}$ where P is an invertible matrix and J is Jordan matrix which has form*

$$J = \mathit{diag}(J_1, ..., J_P)$$

where each block J_i, called Jordan block, is a square matrix of the form

$$J_i = \begin{bmatrix} \lambda_i & 1 & & \\ & \lambda_i & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_i \end{bmatrix}$$

According to its above definition, the power of the matrix M can be computed as follows

$$M^k = (PJP^{-1})^k = PJ^kP^{-1}$$

Hence, M^k converges when $k \to \infty$ if and only if J^k converges. The powers of J is computed via the powers of Jordan block $J_1^k, J_2^k, ..., J_p^k$.

$$J^k = \text{diag}(J_1^k, ..., J_P^k)$$

where

$$J_i^k = \begin{bmatrix} \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \binom{k}{2}\lambda_i^{k-2} & \cdots & \binom{k}{d_i-1}\lambda_i^{k-(d_i-1)} \\ & \lambda_i^k & \binom{k}{1}\lambda_i^{k-1} & \cdots & \binom{k}{d_i-2}\lambda_i^{k-(d_i-2)} \\ & & & \ddots & \\ & & & & \lambda_i^k \end{bmatrix}$$

Note that the powers of J converges if and only if the powers of all Jordan blocks converge. Thus, we focus on the convergence of a block J^k as stated in the following lemma.

Lemma 2. *The convergence of a $d \times d$ Jordan block J_i depends only d and λ_i:*

(1) If $|\lambda_i| > 1$ then J_i^k does not converge when $k \to \infty$.

(2) If $|\lambda_i| < 1$ then J_i^k converges to 0 when $k \to \infty$.

(3) If $|\lambda_i| = 1$ and $\lambda_i \neq 1$, then J_i^k does not converge when $k \to \infty$.

(4) If $\lambda_i = 1$ and $d = 1$, then $J_i^k = \begin{bmatrix} 1 \end{bmatrix}$.

(5) If $\lambda_i = 1$ and $d > 1$, then J_i^k does not converge when $k \to \infty$.

Proof. Cases (1), (3), (4), and (5) are trivial, thus we only show the proof for case (ii). With $|\lambda_i| < 1$, every element of J_i^k has form $\binom{k}{j}\lambda_i^{k-j}$ which converge to 0 as $k \to \infty$. □

According to this lemma, when normalized factors C_s, C_c satisfies $C_s C_c = |\lambda_1|$, we will have

$$\left(\frac{M}{C_s C_c}\right)^{t/2} \mathbf{x}^0 = P \left(\frac{J}{|\lambda_1|}\right)^{t/2} P^{-1}\mathbf{x}^0$$

Clearly, $\mathbf{x}^t$ will converge when $\left(\frac{J}{|\lambda_1|}\right)^{t/2}$ converges. Then, we have the following theorem.

Theorem 3.3. *The centrality vector converges if and only if the characteristic matrix has exactly one maximum magnitude eigenvalue.*

To compute the converged centrality vector, we first choose α such that M has $\lambda_1 > \lambda_2$. In practice, we choose $\alpha = 0.5$ and centrality vectors always converge.

Although it seems necessary to compute the largest eigenvalue of M, we propose an alternative method to avoid this time-consuming computation as follows. Suppose that $\mathbf{x}^{2t}$ converges to a vector $\mathbf{x}$ after t_0 iterations i.e. $\mathbf{x} = \frac{M^{t_0}\mathbf{x}^0}{|\lambda_1|^{t_0}}$. Now we define the sequence of vectors $z_0 = \mathbf{x}^0$, $z_{i+1} = \frac{Mz_i}{|Mz_i|}$, then:

$$z_{t_0} = \frac{M^{t_0} z_0}{\prod_{i=0}^{t_0-1} |Mz_i|} = \frac{M^{t_0}\mathbf{x}^0}{\prod_{i=0}^{t_0-1} |Mz_i|}$$

It means that $\mathbf{x} = Az_{t_0}$ where A is a scalar value. Therefore $z_{t_0} = \frac{\mathbf{x}}{||\mathbf{x}||}$. Thus we can compute the centrality vector using the recursive formula of z as described in Algorithm 6, then use this algorithm as sub-routine to detect critical nodes in Algorithm 7.

Algorithm 6 Iterative Interdependent Centrality

Input: Characterize matrix M and allowed error ϵ
Output: Centrality vector
$\mathbf{x} \leftarrow \mathbf{1}$
$error \leftarrow +\infty$
while $error > \epsilon$ **do**
  $\mathbf{y} \leftarrow M\mathbf{x}$
  $norm \leftarrow ||\mathbf{y}||$
  $\mathbf{y} \leftarrow \mathbf{y}/norm$
  $error \leftarrow ||\mathbf{y} - \mathbf{x}||$
  $\mathbf{x} \leftarrow \mathbf{y}$
end while
Return $\mathbf{x}$

Algorithm 7 IIC-based Algorithm

Input: Interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, an integer k
Output: Set of k critical nodes in $T \in V_s$
$T \leftarrow \emptyset$
for $i = 1$ to k **do**
  $\alpha \leftarrow 0.5$, Compute M
  $\epsilon \leftarrow 10^{-8}$
  Compute centrality vector $\mathbf{x}$ using Algorithm 6.
  $u \leftarrow \text{argmax}_{V_s \setminus T}\, \mathbf{x}[u]$
  $T \leftarrow T \cup \{u\}$
  Update $\mathfrak{I}[V_s \setminus T]$
end for
Return T

Time Complexity: Since two matrices M^s and M^c have only $(2|E_s| + |V_s|)$ and $(|E_c| + |V_c|)$ non-zero elements, the product $M\mathbf{x} = M^s M^c \mathbf{x}$ takes $O(2|E_s| + |V_s| + 2|E_c| + |V_c|)$ time using sparse matrix multiplication. The convergence speed is $\frac{|\lambda_1|}{|\lambda_2|}$, thus the number of iterations is $O(\frac{\log(1/\epsilon)}{\log \frac{|\lambda_1|}{|\lambda_2|}})$. Therefore, the total running time to compute iterative interdependent centrality is $O((|E_s + E_c|)\frac{\log(1/\epsilon)}{\log \frac{|\lambda_1|}{|\lambda_2|}})$. Thus, the total time to detect critical nodes is $O(k(|E_s + E_c|)\frac{\log(1/\epsilon)}{\log \frac{|\lambda_1|}{|\lambda_2|}})$.

3.3.3 Hybrid Algorithm

Motivated by the advantages of Max-Cas and IIC algorithms, we further design a hybrid algorithm by taking advantage of them. As one can see, Max-Cas only works well when networks are loosely connected since it mainly aims to create as many failed nodes as possible instead of making the network as weak as possible. On the other hand, IIC algorithm can make the network weak but it does not work well as Max-Cas when networks are loosely connected. Thus, the idea of hybrid algorithm is to remove as many nodes as possible and make networks weaker in turn. That is, we use Max-Cas and IIC algorithms in odd and even iterations respectively, as described in Algorithm 8. Since the running time of IIC is much smaller than Max-Cas, its running time is about a half of Max-Cas, which will be empirically shown in Section 3.4.

Algorithm 8 Hybrid Algorithm

Input: Interdependent system $\mathfrak{I}(G_s, G_c, E_{sc})$, an integer k
Output: Set of k critical nodes in $T \in V_s$
$T \leftarrow \emptyset$
for $i = 1$ to k **do**
  if i is odd **then**
    Select u as Max-Cas algorithm
  else
    Select u as IIC algorithm
  end if
  $T \leftarrow T \cup \{u\}$
  Update $\mathfrak{I}[V_s \setminus T]$
end for
Return T

3.4 Experimental Evaluation

3.4.1 Dataset and Metric

In the experiment, we evaluate Max-Cas, IIC, and Hybrid algorithms, with respect to the size of giant connected component (GCC) and the running time, on various real-world and synthetic datasets.

In terms of power networks, we use both real Western States power network of the US [55] with 4941 nodes and 6594 edges, and the synthetic scale free networks. This network as well as other communication networks belong to a class of networks called scale-free networks in which the number of nodes with degree d, denoted by P(d), is proportional to $d^{-\beta}$ i.e., P(d) $\sim$ $d^{-\beta}$ for some exponential factor $\beta > 0$. According to [9], power networks are found to have their exponential factors β between 2.5 and 4. In order to do a more comprehensive experiment, we further generate more types of synthetic power networks with different exponential factors, using igraph package [23].

In addition, due to the lack of data describing interdependencies between any communication networks and the real-world power network, we use the synthetic scale-free networks, representing communication networks, e.g. Internet, telephone network, etc. Since most communication networks are observed to have the scale free property with their exponential factors β between 2 and 2.6 [12, 57], we generate communication networks with component factors of 2.2 or 2.6.

For the sake of coupling method, motivated by the observation from real-world interdependent systems in [44], we develop a realistic and practical coupling approach, Random Positive Degree Correlation Coupling (RPDCC) scheme. In this scheme, nodes with high degrees tend to coupled together and so do nodes with low degrees, thus the degree correlation of coupled nodes is positive as described in [44]. The detail of RPDCC strategy will be discussed later in Section 3.5.

Finally, each experiment on synthesized systems is repeated 100 times to compute the average results.

3.4.2 Performance of Proposed Algorithms

In order to show the effectiveness of our proposed algorithms, due to the intractability of IPND problem and the time consumption to obtain optimal solutions, we focus on comparing them with traditional centrality approaches which are often used in network analysis [11], including degree centrality (DC), closeness centrality (CC), betweenness centrality (BC) [13], and bridgeness centrality (BRC) [33]. In these approaches, the k nodes of largest centralities in power networks are selected as critical nodes. Particularly, we test our approaches on the following three types of datasets:

1) WS System: US Western states power network — Scale-free communication network with $\beta = 2.2$.

2) SS System: Scale-free power networks with $\beta = 3.0$ — Scale-free communication network with $\beta = 2.2$.

3) Eq-SS System: Scale-free power and communication networks with the same $\beta = 2.6$.

Here we choose the exponential factor β according to the real-world power networks and communication networks, as mentioned above in 3.4.1.

Fig. 3-3 reports the comparison of performance between different approaches in these three interdependent systems. In these figures, all of three proposed algorithms outperform CC (the best traditional centrality approach) for any number of k critical nodes. When k becomes larger, the interdependent systems have totally destroyed by choosing these critical nodes using our algorithms, while more than 60% of nodes remain intact if selecting nodes with highest traditional centralities. Especially in WS interdependent system consisting of real-world US Western states power system, the number of functional nodes remains nearly 5000 even 50 nodes are identified using CC, whereas it is sufficient to destroy the whole system only by removing 20 nodes using our Hybrid or Max-Cas approach. That is, these traditional approaches perform much worse compared with our algorithms, especially when the number of attacked nodes is large. Thus, the traditional methods cannot identify a correct set of critical nodes

in interdependent systems. The reason is that these approaches can only reflect the importance of each node in single power networks rather than interdependent systems, and they ignore the impact of cascading failures to interdependent systems.

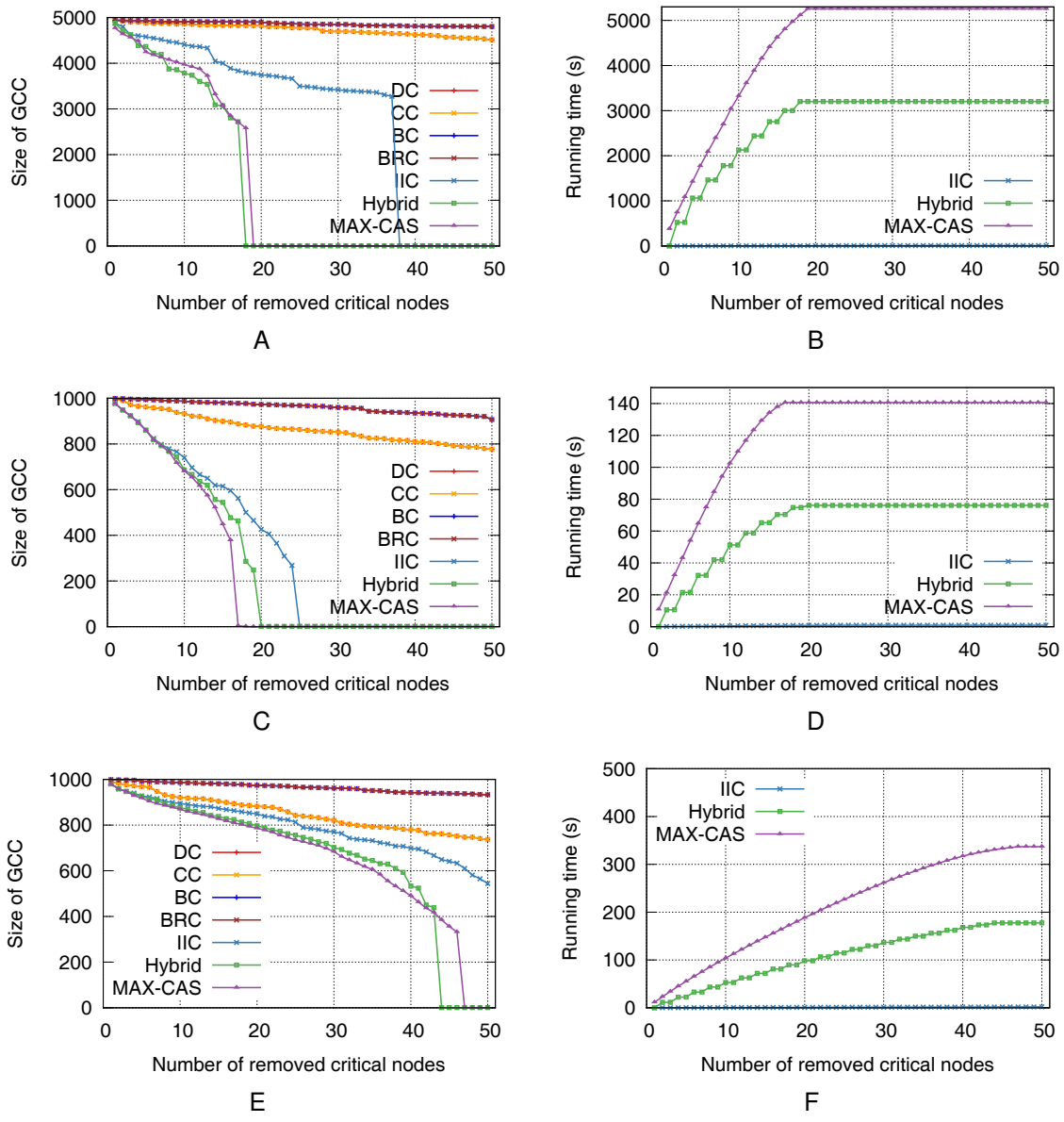


Figure 3-3. Performance Comparison on Different Interdependent Systems: WS System (A, B), SS System (C, D), and Eq-SS System (E, F).

Comparing our three proposed approaches, as revealed in Fig. 3-3, IIC runs fastest in spite of its worst performance, roughly 1000 times faster than Max-Cas in WS interdependent system. We also notice that the performance of Max-Cas and Hybrid

algorithms is very close while Hybrid algorithm runs about 2 times faster than Max-Cas algorithm. In particular, Max-Cas has better performance than Hybrid algorithm in SS interdependent system, yet worse performance in the other two systems. This is because the power network with $\beta = 3.0$ is very loosely connected and fragile in SS interdependent systems. Thus, Max-Cas strategy can destroy the system quickly and easily. However, since nodes are better connected in the other two systems, especially Eq-SS, Hybrid algorithm is more efficient due to its strategy that makes networks weak first and then destroys them. As illustrated in Fig. 3-3E, the performance of Hybrid is lower than Max-Cas initially, but higher than Max-Cas when the networks get weak enough. Additionally, in all of these systems, when the number of removed nodes reach to a certain value, the whole system is failed. These numbers are about 20 for WS and SS system and 40 for Eq-SS system. This shows that interdependent networks are vulnerable, especially when loosely connected.

3.4.3 Vulnerability Assessment of Interdependent Systems

With the effectiveness of Hybrid algorithm observed through the above experiments, we confidently use it to further assess the vulnerability of interdependent systems and explore some insight properties.

3.4.3.1 Different coupled communication networks

We are interested in investigating the vulnerability of a certain power network when it is coupled with different communication networks. First, we fix one synthetic power network by generating a scale-free network with $\beta = 3$ according to [9]. The coupled communication networks are also generated as scale-free networks, with their exponential factors β between 2.5 and 2.7, as mentioned above. All generated networks have 1000 nodes.

As illustrated in Fig. 3-4, the power networks tend to be more vulnerable when their coupled communication networks are more sparse, i.e., with larger exponential factor β. That is, it gives us an intuition that the power networks will become more vulnerable

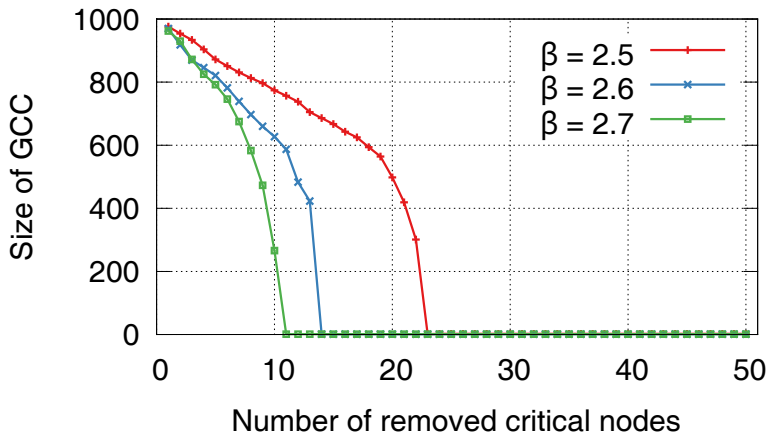


Figure 3-4. The Vulnerability Of A Fixed Power Network

when their coupled networks are easy to be attacked. In particular, in order to destroy the power networks, the numbers of critical nodes in them are 23, 17, and 11 when their coupled communication networks have $\beta = 2.5$, $\beta = 2.6$ and $\beta = 2.7$, respectively, which indicates some key thresholds to protect the function of power networks with the knowledge of their interdependent networks.

3.4.3.2 Disruptor threshold

In this part, we evaluate an important indicator of the vulnerability, the disruptor threshold which is the number of nodes whose removal totally destroys the whole system. The smaller it is, the more vulnerable the system is. We would like to observe the dependence of the disruptor threshold on the network size. Particularly, we generate two scale-free networks with the same size and exponential factors β of 3.0 and 2.2, corresponding to power and communication networks, then couple them using RPDCC scheme.

As shown in Fig. 3-5, the disruptor threshold provided by all proposed algorithms is small and increases slowly with respect to the growth of the network size. When the network size is raised by 5 times, from 1000 to 5000 nodes, the disruptor threshold only increases roughly 3 times. When the size of network is 5000, the disruptor

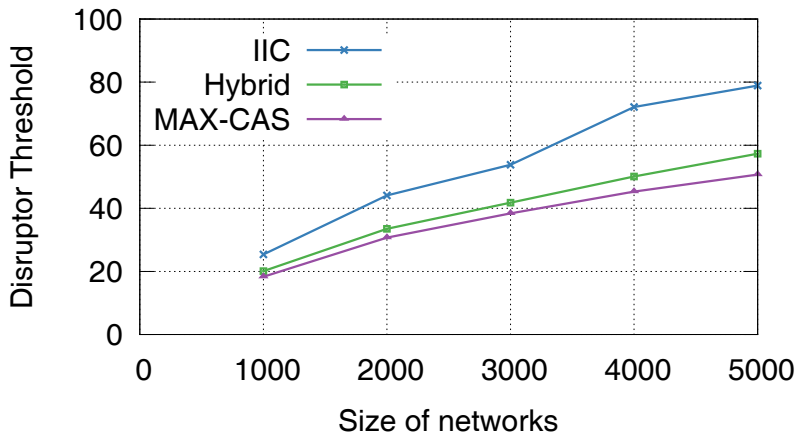


Figure 3-5. The Disruptor Threshold with Different Network Sizes

thresholds of Max-Cas and Hybrid algorithms are roughly 51 and 57. This implies that the removal of 1% number of nodes is enough to destroy the whole system. Even the IIC algorithm needs to destroy only 1.5% fraction of nodes to break the system down. Large interdependent systems seem to be extremely vulnerable under different attack strategies due to the following reason. When the network size grows up, the possibility that a high degree node is dependent on a low degree node also runs up. As a result, it is easier to disable the functionality of high degree nodes which often play an important role in the network connectivity. Therefore, the vulnerability of the interdependent system needs to be reevaluated regularly, especially fast growing up systems.

3.4.3.3 Different coupling schemes

Another interesting observation is to investigate the impacts of the way nodes are coupled to the vulnerability of interdependent system. Apart from the RPDCC scheme, we evaluate the robustness with other three coupling strategies, as follows:

1) *Same Degree Order Coupling (SDOC)*: The nodes of i^{th} highest degree in two networks are coupled together.

2) *Reversed Degree Order Coupling (RDOC)*: The node of i^{th} highest degree in one network is coupled with the node of i^{th} lowest degree in the other network.

3) *Random Negative Degree Correlation Coupling (RNDCC)*: A node of higher degree in one network are randomly, with lower probability, to couple with another node of higher degree nodes in the other network.

Note that the RNDCC scheme is the opposite strategy to the RPDCC scheme (in Section 3.5). We test on the interdependent systems, consisting of a power network with $\beta = 3$ and a communication network with $\beta = 2.2$ using the four different coupling schemes. All networks have 1000 nodes.

Fig. 3-6 reports the vulnerability of power networks when coupling them with communication networks in different manners. As one can see, SDOC provides the most robust interdependent system, although it is not practical. The size of the remained giant connected component decreases slowly when the number of removed nodes increases. On the other hand, RDOC makes the system very vulnerable, which can be destroyed by only removing 1 nodes from the power network. This is because the nodes of lower degree in communication networks are very easy to be failed, which, immediately, cause the failures to their coupled nodes of higher degree in power networks. When many high degree nodes are removed, the network is easy to be fragmented which leads to the destruction of the whole system shortly. The interdependent systems with the other two schemes, RPDCC and RNDCC, illustrate their robustness between those using SDOC and RDOC, due to the random factors in RPDCC and RNDCC. Compared with RNDCC, systems coupled by RPDCC is almost twice more robust because of its positive correlations. These results point out an important principle that the higher correlation between the degrees of coupled nodes, the stronger the interdependent system is. In other words, a node of high degree in one network should not be coupled with a node of low degree in the other network; otherwise, this node will be a weak point to attack.

3.5 RPDCC / RNDCC Coupling Schemes

In this section, we present the RPDCC scheme to randomly couple two networks with positive degree correlation. Given two network G_s and G_c, we form two weighted

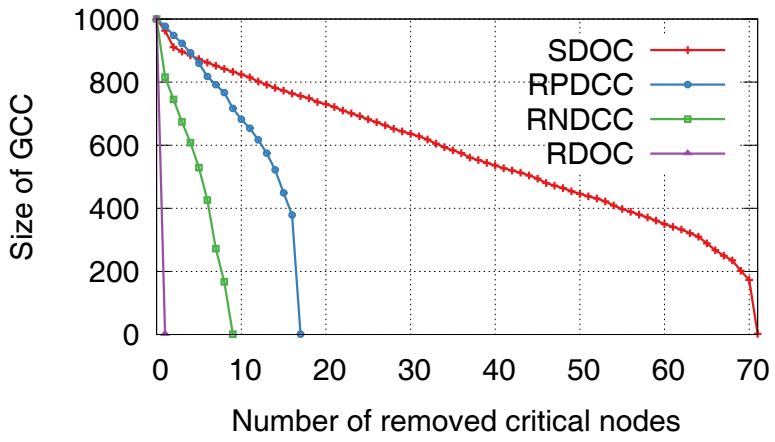


Figure 3-6. Vulnerability Comparison using Different Coupling Schemes

Algorithm 9 Random weighted permutation

Input: A weighted set of n elements $X = \{x_1, x_2, \dots, x_n\}$ with weights $w(\cdot)$
Output: Weighted random permutation Y of X.
$total \leftarrow \sum_{i=1}^{n} w(x_i)$
for $i = 1$ to n **do**
  $e \leftarrow$ a random selected element in X with probability $w(e)/total$
  $Y[i] \leftarrow e$; $X \leftarrow X \setminus \{e\}$; $total \leftarrow total - w(e)$
end for
Return Y

sets that contain vertices of G_s and G_c as elements and their degrees as weights. Then we generate two random weighted permutations $\{v_1^{s'}, v_2^{s'}, \dots, v_n^{s'}\}$ and $\{v_1^{c'}, v_2^{c'}, \dots, v_n^{c'}\}$ of nodes in G_s and G_c as described in in Algorithm 9, then $v_i^{s'}$ is coupled with $v_i^{c'}$, $1 \leq i \leq n$. In the following theorem, we show that a node of larger weight has smaller expected index in each permutation, that is, nodes of high degrees in two permutations tend to have low indices. In other words, this results in the positive correlation between degrees of coupled nodes. (For RNDCC, we couple $v_i^{s'}$ with $v_{n-i}^{c'}$.)

Theorem 3.4. *In the random weighted permutation, an element with bigger weight has lower expected index than an element with smaller weight.*

Proof. Let $\mathsf{E}(X, e)$ be the expected index of an element e in the random weighted permutation. Then, we have:

$$\mathsf{E}(X, e) = \frac{w(e)}{\sum_{x \in X} w(x)} + \sum_{z \in X \setminus \{e\}} \frac{w(z)}{\sum_{x \in X} w(x)} (1 + \mathsf{E}(X \setminus \{z\}, e)$$

Therefore, $\mathsf{E}(X, e_1) \leq \mathsf{E}(X, e_2)$ if $w(e_1) \geq w(e_2)$. □

3.6 Related Works

Most of works on network vulnerability assessment are studied in single networks [8, 10, 21, 29, 41]. The centrality measurements [11] are widely used, including degree, betweenness and closeness centralities, average shortest path length [5], global clustering coefficients [39]. Alternatively, Arulselvan *et al.* [8] first proposed the total pairwise connectivity as an effective measurement, based on which they propose the CND problem and designed a heuristic to detect critical nodes. The β-disruptor problem was later defined by Dinh *et al.* [25] followed by pseudo-approximation algorithms. Unfortunately, these approaches fail to accurately identify the critical nodes on interdependent networks.

Recently, vulnerability assessment of interdependent networks was initiated by Buldyrev *et al.* [15], and followed by a set of related papers [15, 27, 32, 45, 48]. These works validated the size of largest connected component as an effective metric for cascading failures, covering a wide range of the random failures [15], order percolation phase transition [16, 27, 45] and exploitation of robustness under targeted attacks [32]. Their results illustrated that interdependent networks are much more vulnerable than single networks. Unfortunately, these works heavily depend on configuration models and, therefore, not applicable to real-world networks. And none of them proposed a strategy to identify top critical nodes in interdependent networks.

3.7 Summary

In this chapter, we studied the optimization problem of detecting critical nodes to assess the vulnerability of interdependent power networks based on the well-accepted

cascading failure model and metric, the size of largest connected component. We showed its NP-hardness, along with its inapproximability. Due to its intractability, we proposed a greedy framework with various novel centralities, which measures the importance of each node more accurately on interdependent networks. The extensive experiment not only illustrates the effectiveness of our approaches in networks with different topologies and interdependencies, but also reveals several important observations on interdependent power networks.

CHAPTER 4
INFLUENCE DIFFUSION IN MULTIPLE ONLINE SOCIAL NETWORKS

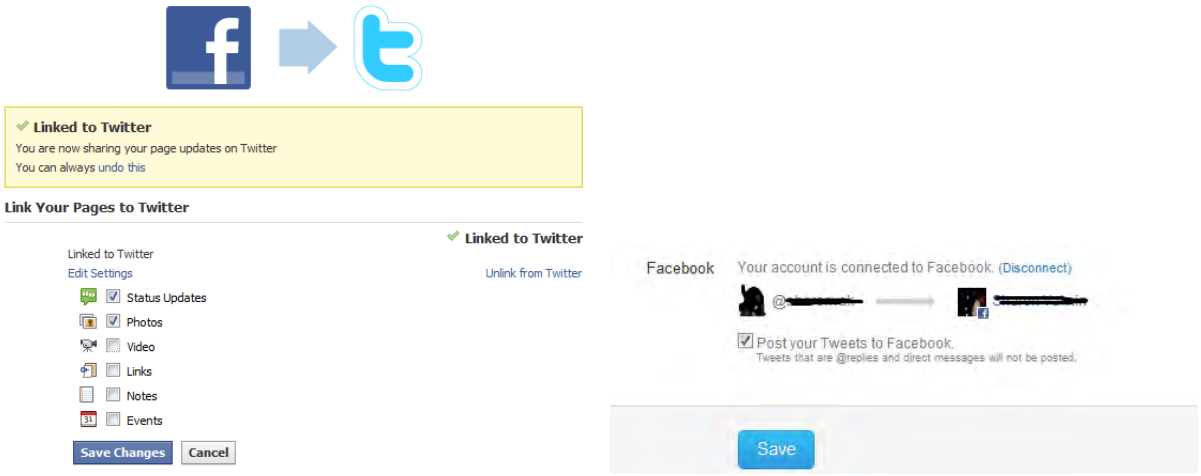
In the recent decade the popularity of online social networks, such as Facebook, Google+, Myspace and Twitter etc., has created a new major communication medium and formed a promising landscape for information sharing and discovery. On average, Facebook users spend 7h:45 per person per month [4]; 3.2 billion likes and comments are posted every day on Facebook [3]; 340 million tweets are sent out everyday on Twitter [4]. Such engagement of online users fertilizes the land for information propagation to a degree never achieved before in mass media. More importantly, OSNs also inherit one of the major properties of real social networks- the word-of-mouth or peer-pressure effect in which an individual's opinion or decision is influenced by his friends and colleagues. Due to the considerable impact of this effect on the popularity of new products [14, 28], OSNs have rapidly become one of the most attractive choices to rising the awareness of new products or brands as well as to reinforce the connection between customers and companies. The crucial problem is how to find the smallest set of influencers who can influence a massive number of users.

There is a considerable number of overlapping users among multiple OSNs which creates a huge effect on the diffusion of information in these networks. When a user joins multiple networks, s/he can relay the information from one network to another. Let us consider the following typical scenario to illustrate this phenomenon. Jack, a user of both Twitter and Facebook, logs in Twitter and knows about an excellent product from his friend. He right away falls in love with the new product and eagerly shares the information by tweeting it. Moreover, he configured his Twitter and Facebook accounts as illustrated in Fig. 4-1 that allows him to automatically post on his Facebook's wall whenever he has a new tweet and vice versa. As the consequence, the product information is exposed to his friends in both networks and the information further spreads out on both the networks. If we only consider the information propagation in one

network, the reach of the information is estimated incorrectly and thus the influence of users in these networks. In this case, the influence of Jack should be the combination of his influence in both networks. As shown in Fig. 4-2, the fraction of overlapping users is considerable, therefore studying the problem only in one network provides a solution which is quite different from reality. This provides the motivation to study the above problem on multiple networks where the influence of users is evaluated based on multiple OSNs in which they participate.

Nearly all the existing works studied different variants of the massive influence problem on a single network [18, 19, 34, 35, 37, 52, 61, 62]. Kempe et al. [34] first formulated the influence maximization problem which asks to find a set of k users who can maximize the influence. The influence is propagated based on a stochastic process called Independent Cascade Model (IC) in which a user will influence his friends with probability proportional to the strength of their friendship. The author proved that the problem is NP-hard and proposed a greedy algorithm with approximation ratio of $(1 - 1/e)$. After that, a considerable number of works study and design new algorithms for the problem variants on the same or extended models such as [18, 35]. There are also works on the linear threshold (LT) model for influence propagation in which a user will adopt the new product when the total influence of his friends surpass some threshold. Feng et al. [62] showed NP-completeness for the problem and Dinh et al. [24] proved the inapproximability as well as proposed efficient algorithms for this problem on a special case of LT model. In their model, the influence between users are uniform and a user is influenced if a certain fraction ρ of his friends are active.

Recently, researchers have started to explore multiple networks with works of Yagan et al. [58] and Liu et al. [38] which study the connection between offline and online networks. The first work investigates the outbreak of information using the SIR model on random networks. The second one analyzes networks formed by online interactions and offline events. The authors focus on understanding the flow of information and



A Auto post from Facebook to Twitter

B Auto post from Twitter to Facebook

Figure 4-1. Auto update across social networks

users of these sites...

...are also users of these sites	LinkedIn	Facebook	Twitter	MySpace
LinkedIn	-	12%	21%	6%
Facebook	82%	-	91%	57%
Twitter	31%	20%	-	17%
MySpace	36%	49%	70%	-

Figure 4-2. The number of shared users between major OSNs in 2009 [2]

network clustering but not solving the massive influence problem. But these works do not study any specific optimization problem of viral marketing. Shen et al. [49] explore the information propagation on multiple online social networks taking into account the interest and engagement of users. In their solution, all networks are combined into one network by representing an overlapping user as a super node. This method cannot preserve the individual networks' properties.

In this chapter, we study the Massive Influence problem (MIP) which asks for a set of users with minimum cardinality to influence a certain fraction of users in multiple networks. Suppose that we know the participant of users overall networks, we exploit the additional information of overlapping users to identify top influent ones over multiple

networks. Although the problem has been studied in a single network but those are only special cases such as uniform influence between users in [24], etc. In addition, the overlapping users introduce several new challenges, so the previous solutions cannot be easily adopted. For example, how to evaluate the influence of overlapping users across multiple networks? In which network, a user is easier to be influenced? We introduce novel coupling schemes which combine multiple networks into one network while retaining the influential properties of the original networks partially or fully. After coupling the networks, we can exploit existing solutions on the single network to solve the problem. This is a powerful and comprehensive procedure to study MIP. Moreover, we propose a new metric called influence relay to analyze the flow of influence between networks. Through comprehensive experiments, we discover crucial properties of the multiple networks in diffusing the information.

The chapter is organized as follows. In Section 4.1, we present the influence propagation model on multiple network and define the problem. We then introduce the method to align nodes in networks in Section 4.2. After that, we introduces different coupling schemes in Section 4.3 and 4.4. We next present the influence relay to study the influence propagation process in Section 4.5. Section 4.6 shows experimental results. In addition, we present coupling schemes for two stochastic cascading models in Section 4.7. Finally, we summarize the chapter in Section 4.8.

4.1 Network Model and Problem Definition

4.1.1 Graph Notations

We consider k networks $G^1, G^2, \dots, G^k$, each of which is modeled as a weighted directed graph $G^i = (V^i, E^i, \theta^i, W^i)$. The vertex set $V^i = \{u\text{'s}\}$ represents the participation of $n^i = |V^i|$ users in the network G^i, and the edge set $E^i = \{(u, v)\text{'s}\}$ contains $m^i = |E^i|$ oriented connections (e.g., friendships or relationships) among network users. $W^i = \{w^i(u, v)\text{'s}\}$ is the (normalized) weight function associated to all edges in the i^{th} network. In our model, weight $w^i(u, v)$ can also interpreted as the

strength of influence (or the strength of the relationship) a user u has on another user v in the i^{th} network. The sets of incoming and outgoing neighbors of vertex u in network G^i are denoted by N_u^{i-} and N_u^{i+}, respectively. In addition, each user u is associated with a threshold $\theta^i(u)$ indicating the persistence of his opinions. The higher $\theta^i(u)$ is, the more unlikely that u will be influenced by the opinions of his friends. Furthermore, the users that actively participate in multiple networks are referred to as *overlapping users*. Those users are considered as bridge users for information propagation across networks. Finally, we denote by $G^{1...k}$ the system consisting of k networks, and by U the exhaustive set of all users $U = \cup_{i=1}^{k} V^i$.

4.1.2 Influence Propagation Model

We first describe the *linear threshold model* (LT-model) [24, 62], a popular model for information and influence diffusion in a single network, and then discuss how this LT model can be extended to cope with multiple networks. In the original LT model, each network user u is either in an *active* or *inactive* state: u is in an *active* state if he originally adopts the information, or the total influence from his direct neighbors exceeds his threshold $\theta(u)$, i.e, $\sum_{v \in N(u)} w(v, u) \geq \theta(u)$. Otherwise, u is in an *inactive* state.

In a big picture, given a system of k networks, the information is propagated separately in each network and can only be transferred from one network to another via the overlapping users of these networks The information starts to spread out from set of seed users S i.e. all users in S have active state and the remaining users are inactive. At time t, a user u becomes active if the total influence from its active neighbors surpasses its threshold in some network i.e. there exist i such that:

$$\sum_{v \in N_u^{i-}, v \in A} w^i(v, u) \geq \theta^i(u)$$

where A is the set of active users after time $(t-1)$.

After each time step, new inactive users are activated and they continue to activate other users. The process will continue until there are no more inactive users can be

activated. If we limit the propagation time to d, then the process will stop after $t = d$ time steps. The set of active users caused by the seed set S after time d is denoted as $A^d(G^{1...k}, S)$. Note that d is also the number of hops in networks up to which the influence can be propagated from the seed set, so d is called the number of propagation hops.

4.1.3 Problem Definition

In this chapter, we address the fundamental vulnerability problem on multiple networks: the **Massive Influence** problem. The problem asks to find a seed set of minimum cardinality which influences a large fraction of users, formally defined as follows.

Definition 4. *(Massive Influence Problem (MIP)) Given a system of k networks $G^{1...k}$ with the set of users U, a positive integer d, and $0 < \beta \leq 1$, the MIP problem asks to find a seed set $S \subset U$ of minimum cardinality such that the number of active users after d hops according to LT model is at least β fraction of users i.e. $|A^d(G^{1...k}, S)| \geq \beta|U|$.*

When $k = 1$, we have the variant of the problems on a single network which NP-hard to solve [17] but it is easier to design heuristic algorithm on the single network. In next sections, we present different coupling strategies to reduce the problem on multiple networks to one on a single network in order to utilize the algorithm designing.

4.2 Network Alignment

We first reassign a universal identification (id) to each node in the networks such that all overlapping nodes of the same user have the same id. Each network topology often uses its own system for naming nodes, thus a person may have different ids in different networks. As a consequence, it requires a complicated mechanism and extra effort repeatedly to keep updating the user states across networks. We ease this burden by assigning a unique id to each user and using it as the node id in all networks. However, if we trivially assign new id to an unassigned node and its overlapping nodes one by one, we need to scan all the overlap mappings each time which is almost

impractical in large networks. Thus, we need to design an algorithm which assigns new ids in only linear time.

Our goal is to scan each overlap mapping and check each node once to assign new ids. Instead of assigning ids to all overlapping nodes of a user at the same time, we check whether one of its overlapping nodes is already assigned an id or not. To be specific, we process each network in two phases: assign ids to nodes in its mapping lists with the processed networks and then assign new ids to the remaining nodes. This method guarantees the validity of new ids as stated in Lemma 3. The algorithm, as described in Algorithm 10, scans each mapping and checks each node once, so the total running time is linear in the total size of networks and overlap mappings.

Algorithm 10 Node-Alignment Algorithm

Require: k networks $G^{1...k}$ and overlap mappings $\{C_{ij}\}$.
Ensure: A new id mapping for nodes in $G^{1...k}$.

```
1:  newid ← 0
2:  Initialize id mapping M
3:  for i = 1 to k do
4:      for j = 1 to i − 1 do
5:          for each pair (u, v) ∈ C_ij do
6:              M[u] = M[v]
7:          end for
8:      end for
9:      for each u ∈ V^i do
10:         if u is unassigned then
11:             M[u] = newid
12:             newid ← newid + 1
13:         end if
14:     end for
15: end for
16: Return M
```

Lemma 3. *Node-Alignment Algorithm assigns the same id to all overlapping nodes of the same user and different ids to nodes of different users.*

Proof. Let $u^{i_1}, u^{i_2}, \ldots, u^{i_l}$ be overlapping nodes of user u in networks $i_1 < i_2 < \ldots < i_l$, then $\mathcal{M}[u^{i_j}] = \mathcal{M}[u^{i_{j-1}}] = \ldots = \mathcal{M}[u^{i_1}]$ due to the line 6 of the algorithm. Thus, all overlapping nodes of user u have the same id.

Next, consider two overlapping nodes u^i and v^j of users u and v. Denote u^{i_0} (v^{j_0}) as overlapping nodes of u (v) in network i_0 (j_0) with the smallest index. Without loss of generality, suppose that u^{i_0} is assigned an id after v^{j_0}. Due to the property of u^{i_0}, it does not appear in any overlap mapping with previous networks, hence it is assigned with a new id. Thus, u^{i_0} and v^{j_0} have different ids; so are u^i and v^j. □

4.3 Lossless Coupling Schemes

In this section, we present two schemes to couple multiple networks into a new single network with respect to the influence diffusion process on each participant network. A notable advantage of this newly coupled graph is that we can use any existing algorithm on a single network to produce the solution on multiple networks with the same quality. Unfortunately, we encounter a series of the challenging issues in designing such coupling schemes:

(1) The heterogeneity of user participation. A user may join in one, two, or more networks. How can we recognize and differentiate these users in the coupled network? How to capture the roles of each user on the multiple networks?

(2) The process of information and influence propagation among networks. In multiple networks, when a user is influenced he tends to immediately propagate the information on all networks that he is a part of. How can we describe this immediate transmission of the information between networks in just a single network?

(3) Preserving properties of individual networks. The coupled network should be a good representative of all the individual networks. It should preserve the diffusion properties of all the networks. That enable us to establish the relationship between solution of the problem on the coupled network and the original networks. How can we design a coupling scheme that addresses this issue?

Next, we present the method to overcome these challenges. The first issue is solved by introducing dummy nodes for each user u in networks that it does not belong

to. These dummy nodes are isolated. Now the vertex set V^i of i^{th} network can be represented by $V^i = \{u_1^i, u_2^i, \dots, u_n^i\}$ where $U = \{u_1, u_2, \dots, u_n\}$ is the set of all users. In the new representation, there is an edge from u_p^i to u_q^i if u_p and u_q are connected in G^i. Now we can union all k networks to form a new network G. The approach to overcome the second challenge is to allow nodes $u^1, u^2, \dots, u^k$ of an user u to influence each other e.g. adding edge (u^i, u^j) with weight $\theta(u^j)$. When u^i is influenced, u^j is also influenced in the next time step as they are actually a single overlapping user u, thus the information is transferred from network G^i to G^j. But an emerged problem is that the information is delayed when it is transferred between two networks. Right after being activated, u^i will influence its neighbors while u^j needs one more time step before it starts to influence its neighbors. It would be better if both u^i and u^j start to influence their neighbors in the same time. For this reason, new gateway node u^0 is added to G such that both u^i and u^j can only influence other nodes through u^0. In particular, all edges (u^i, v^i) $((u^j, z^j))$ will be replaced by edges (u^0, v^i) $((u^0, z^j))$. In addition, more edges are added between u^0, u^i, and u^j to let them influence each other. We describe the coupling schemes next and how we can couple the multiple networks preserving their individual properties.

4.3.1 Clique Lossless Coupling Scheme

Given k networks G^1, G^2, ..., G^k with the set of users U, we construct a new graph $G = (V, E, \theta, w)$ as follows.

Firstly, we add dummy vertices of threshold 1 to all these networks and include all nodes into vertex set V together with gate way nodes $V = \cup_{i=1}^{k} V^i \cup \{u_1^0, u_2^0, \dots, u_n^0\}$. In the new vertex set, $u_1^0, u_2^0, \dots, u_n^0$ represents the set of users in the coupled network and are called *user vertices*. Vertex u_p^i called the *account vertex* of user u_p in G^i. The thresholds of the former type of vertices are set to $\theta(u_p^0) = 1$, $1 \leq p \leq n$, and the thresholds of the later type vertices are kept the same with the one in multiple networks i.e. $\theta(u_p^i) = \theta^i(u_p)$.

Secondly, we represent the influence of user u on user v in network G^i by the influence of user vertex u^0 on the account vertex of v in G^i. It means that if there is an edge between user u and v in G^i, then an edge from u^0 to v^i with weight $w(u^0, v^i) = w^i(u, v)$ is added to the edge set E.

Finally, we connect user vertex and account vertices of the same user to guarantee that they have same activation states. The goal is that if one of these nodes is active, it will activate all other nodes. It can be done by adding extra edges called synchronization edges between these nodes whose weights equal to the thresholds of destination nodes. Specifically, $w(u^i, u^j) = \theta(u^j), \forall 0 \leq i, j \leq k, i \neq j$. These synchronization edges form a clique between nodes, thus this coupling scheme is named *clique lossless coupling scheme*. A simple example of the scheme is illustrated in Figure 4-3.

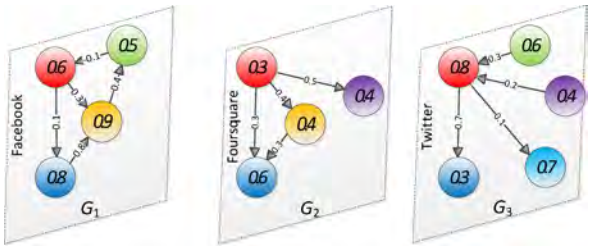
Next we will show that the propagation process in the original multiple networks and the coupled network is actually the same. Influence is alternatively propagated between user and account vertices, so problem with d hops in the multiple networks is equivalent to problem with $2d$ hops in the coupled network.

Lemma 4. *Suppose that the the propagation process on the coupled network G starts from the seed set which contains only user vertices $S = \{s_1^0, \dots, s_p^0\}$, then user vertices is only activated in even propagation hops.*

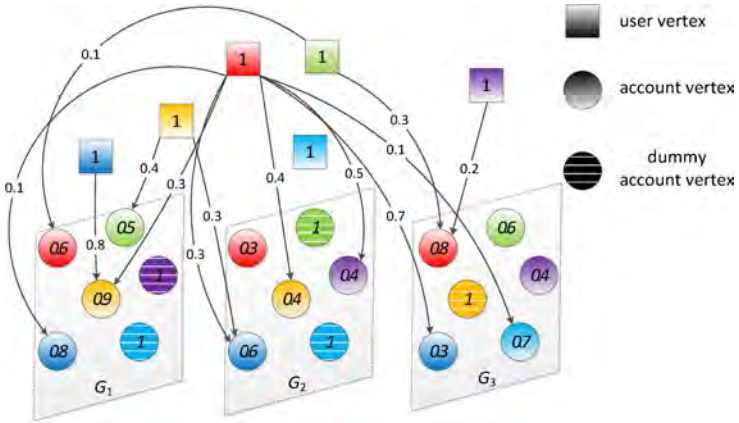
Proof. Suppose that a user vertex u^0 is the first user vertex that is activated at the odd hops $2d + 1$. u^0 must be activated by some vertex u^i and u^i is the the first activated vertex among vertices $u^1, u^2, \dots, u^k$. It means that u^i is activated in hop $2d$. Since all incoming neighbors of u^i is user vertices, some user vertex changes its status to active in hop $2d - 1$. It is contradicted. □

Lemma 5. *Suppose that the the propagation process on $G^{1\dots k}$ and G starts from the same seed set S, then following conditions are equivalent:*

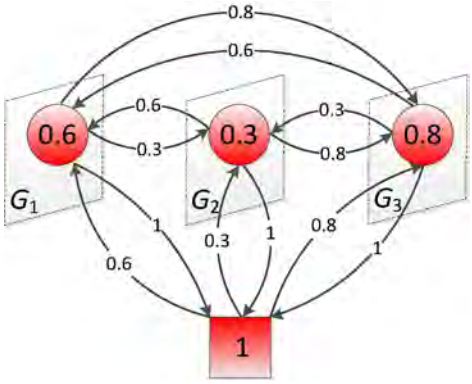
(1) User u is active after d propagation hops in $G^{1\dots k}$.



A Multiple networks G^1, G^2, G^3 with 6 users where vertices with the same color represent the same user. Each user may have different thresholds in different networks, e.g., red user has thresholds of 0.6, 0.3, and 0.8.



B The influence between users in multiple networks are encoded by the influence from user vertices to account vertices. Dummy account vertices are added to guarantee that all users have the same number of account vertices.



C Clique Synchronization

Figure 4-3. An example of *lossless coupling scheme*

(2) There exists i such that u^i is active after $2d-1$ propagation hops in G.

(3) Vertex u^0 is active after $2d$ propagation hops in G.

Proof. We will prove this lemma by induction. Suppose it is correct for any $1 \leq d \leq t$, we need to prove it is correct for $d = t+1$. Denote $A^{1\dots k}(t)$ and A(t) as the set of active users and active vertices after t propagation hops in $G^{1\dots k}$ and G, respectively.

$(1) \Rightarrow (2)$: If user u is active at time $t+1$ in $G^{1\dots k}$, it must be activated at some network G^j. We have:

$$\sum_{v \in N_u^{j-} \cap A^{1\dots k}(t)} w^j(v,u) \geq \theta^j(u)$$

Due to the induction assumption, for each $v \in A^{1\dots k}(t)$, we also have $v^0 \in A(2t)$ in G. Thus:

$$\sum_{v^0 \in N_{u^j}^- \cap A(2t)} w(v^0, u^j) = \sum_{v \in N_u^{j-} \cap A^{1\dots k}(t)} w^j(v,u) \geq \theta^j(u) = \theta(u^j)$$

It means that u^j is active after $(2(t+1)-1)$ propagation hops.

$(2) \Rightarrow (3)$: If there exists i such that u^i is active after $2(t+1)-1$ propagation hops on G, then u^i will activate u^0 in hop $2(t+1)$

$(3) \Rightarrow (1)$: Suppose that $u^0 \notin S$ is active after $2(t+1)$ propagation hops in G, then there must exists u^j which activates u^0 before. This is equivalent to:

$$\sum_{v \in N_{u^j}^-, v \in A(2t)} w(v, u^j) \geq \theta(u^j)$$

For each $v \in A(2t)$, we also have $v \in A^{1\dots k}(t)$. Replace this into the above inequality we have:

$$\begin{aligned}\sum_{v \in N_u^{j-} \cap A^{1\dots k}(t)} w^j(v,u) &= \sum_{v^0 \in N_{u^j}^- \cap A(2t)} w(v^0, u^j) \\ &\geq \theta(u^j) = \theta^j(u)\end{aligned}$$

Thus, u is active in network G^j after $t+1$ propagation hops. □

Next, we will show that the number of influenced vertices in the coupled networks is always $(k+1)$ times the number of influenced users in multiple networks as stated in Theorem 4.1.

Theorem 4.1. *Given a system of k networks $G^{1...k}$ with the user set U, the coupled network G produced by the lossless coupling scheme, and a seed set $S = \{s_1, s_2, \dots, s_p\}$, if $A^d(G^{1...k}, S) = \{a_1, a_2, \dots, a_q\}$ is the set of active users caused by S after d propagation hops in multiple networks, then $A^{2d}(G, S) = \{a_1^0, a_1^1, \dots, a_1^k, \dots, a_q^0, a_q^1, \dots, a_q^k\}$ is the set of active vertices caused by S after $2d$ propagation hops in the coupled network.*

Proof. For each user $a_i \in A^d(G^{1...k}, S)$ i.e. a_i is active after d hops in $G^{1...k}$, then there exists a_i^j which is active after $2d-1$ hops in G according to the Lemma 5. As a consequence, all $a_i^0, a_i^1, \dots, a_i^k$ are active after $2d$ hops. So $B = \{a_1^0, a_1^1, \dots, a_1^k, \dots, a_q^0, a_q^1, \dots, a_q^k\} \subseteq A^{2d}(G, S)$.

Let consider a vertex of $A^{2d}(G, S)$ which is:

Case 1. A user vertex u^0 which is active after $2d$ hops in G, so vertex u must be active after d hops in $G^{1...k}$. This implies $u \in A^d(G^{1...k}, S)$, thus $u^0 \in B$.

Case 2. An account vertex u^i. If u^i is active after $2d-1$ hops, then u must be active after d hops due to Lemma 5, thus $u \in A^d(G^{1...k}, S)$. Otherwise, u^i is activated at hop $2d$, it must be active by some vertex u^j, $j > 0$ since all user vertices only change their state at even hops. Again, $u \in A^d(G^{1...k}, S)$. This results in $u^i \in B$.

From two above cases, we also have $A^{2d}(G, S) \subseteq B$. So that $A^{2d}(G, S) = B$, the proof is completed. □

Theorem 4.1 provides the basis to derive the solution for MIP on multiple networks from the solution on a single network. It implies an important algorithmic property of the *lossless coupling scheme* regarding to the relationship between the solutions of MIP in $G^{1...k}$ and G. The equivalence of two solutions is stated below:

Theorem 4.2. *When the lossless scheme is used, the set $S = \{s_1, s_2, \dots, s_p\}$ influences β fraction of users in $G^{1\dots k}$ after d propagation hops if and only if $S' = \{s_1^0, s_2^0, \dots, s_p^0\}$ influences β fraction of vertices in coupled network G after $2d$ propagation hops.*

Size of the coupled network. The size of the coupled network can be computed from the sizes of the original networks as follows:

Proposition 4.1. *When the lossless scheme is used, the coupled network has $|V| = (k+1)|U| = (k+1)n$ vertices and $|E| = \sum_{i=1}^{k} |E^i| + nk(k+1)$ edges.*

Proof. In the coupling scheme, each user u has $k+1$ corresponding vertices $u, u^1, \dots, u^k$ in the coupled network, thus the number of vertices is $|V| = (k+1)|U| = (k+1)n$. The number of edges equals the total number of edges from all input networks plus the number of new edges for synchronizing. Thus the total number of edges is $|E| = \sum_{i=1}^{k} |E^i| + nk(k+1)$. □

4.3.2 Star Lossless Coupling Scheme

In *clique lossless coupling scheme*, the number of edges to synchronize the state of vertices $u^0, u^1, \dots, u^k$ is $k(k+1)$ for each user u, which results in $nk(k+1)$ extra edges in the coupled network. In real networks, the number of edges is often linear to the number of vertices, so the number of extra edges considerably increases the size of the coupled network, especially when k is large. We would like to design a synchronization strategy that reduces these extra edges.

Note that the large number of extra edges is due to the direct synchronization between every pairs of account vertices of u in *clique lossless coupling scheme*, so we can save some edges by using indirect synchronization. We create one more intermediate vertex u^{k+1} with threshold $\theta(u^{k+1}) = 1$ and let the active state propagate from any vertex in $u^1, u^2, \dots, u^k$ via this vertex. Specifically, the synchronization edges are established follows: $w(u^i, u^{k+1}) = 1$ and $w(u^{k+1}, u^i) = \theta(u^i)$ $1 \le i \le k$; $w(u^{k+1}, u^0) = w(u^0, u^{k+1}) = 1$. The synchronization strategy of *star lossless coupling scheme* is

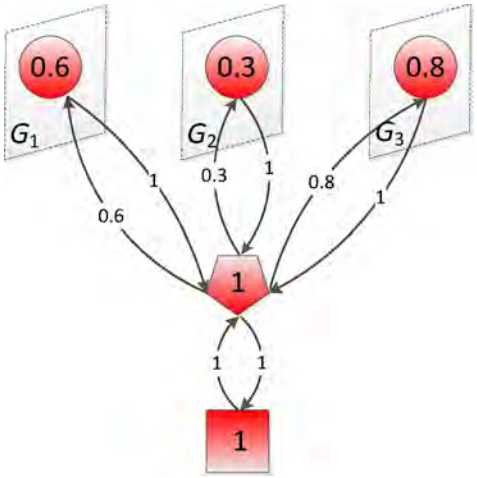


Figure 4-4. Star Synchronization

illustrated in Fig. 4-4. Now, the number of extra edge for each user is $2(k+1)$ and the size of the coupled network is reduced to:

Proposition 4.2. *When star lossless scheme is used, the coupled network has* $|V| = (k+2)|U| = (k+2)n$ *vertices and* $|E| = \sum_{i=1}^{k} |E^i| + 2n(k+1)$ *edges.*

In *star lossless coupling scheme*, it takes 2 hops to synchronize the states of account vertices of each user which leads to delaying the propagation of influence in the coupled network. Due to the similarity between *star lossless scheme* and *clique lossless scheme*, we state the following property of *star lossless scheme* without proof.

Theorem 4.3. *When star lossless coupling scheme is used, the set* $S = \{s_1, s_2, \ldots, s_p\}$ *influences* β *fraction of users in* $G^{1\ldots k}$ *after* d *propagation hops if and only if* $S' = \{s_1^0, s_2^0, \ldots, s_p^0\}$ *influences* β *fraction of vertices in coupled network* G *after* $3d$ *propagation hops.*

4.4 Lossy Coupling Schemes

In the preceding coupling schemes, a complicated coupled network is produced with a large number of auxiliary vertices and edges. It is ideal to have a coupled network which only contain users as nodes. This network provides a compact view of the relationship between users crossing the whole system of networks. To compact the

information which is completely described by the whole system into one network, the loss of information is unavoidable. The goal is to design a scheme such that minimize the loss as much as possible i.e. the solution for the problem in the coupled network is very closed to one in the original system. Next, we present such scheme based on the following key observations.

Observation 1. Consider user u, u will be activated if there exists i such that:

$$\sum_{v^i \in N^-_{u^i}, v \in A} w^i(v^i, u^i) \geq \theta^i(u)$$

where A is the set of active users.

We can relax the condition to activate u with positive parameters $\alpha^1(u)$, $\alpha^2(u)$, ..., $\alpha^k(u)$ as follows:

$$\sum_{i=1}^{k} (\alpha^i(u) \sum_{v^i \in N^-_{u^i}, v \in A} w^i(v, u)) \geq \sum_{i=1}^{k} \alpha^i(u)\theta^i(u^i) \tag{4–1}$$

Proposition 4.3. *Given a system of networks $G^{1...k}$, if the condition (4–1) is satisfied, then user u is activated.*

Proof. When the condition is satisfied, there must exist i such that $\alpha^i(u) \sum_{v^i \in N^-_{u^i}, v \in A} w^i(v, u) \geq \alpha^i(u)\theta^i(u)$. As a consequence, the condition to activate u is satisfied since $\alpha^i(u) > 0$ □

Note that sometimes the condition to activate u is met, but the condition (4–1) is still need more influence from u's friends to satisfy. The more this extra influence need is, the looser condition (4–1) is. We can reduce this redundancy by increasing the value of $\alpha^i(u)$ proportional to the value of $\sum_{v^i \in N^-_{u^i}, v \in A} w^i(v, u) - \theta^i(u)$. In the special case, if $\sum_{v^i \in N^-_{u^i}, v \in A} w^i(v, u) > \theta^i(u)$ and we choose $\alpha^i(u) \gg \alpha^j(u), \forall j \neq i$, then there is no redundancy. Unfortunately, we do not know before hand in which network user u will be activated, so we can only choose parameters heuristically.

Observation 2. When user u participates in multiple networks, it is easier to influence u in some network than the others. The following simple case illustrate such situation. Suppose that we have two networks.In network 1, $\theta^1(u^1) = 0.1$ and

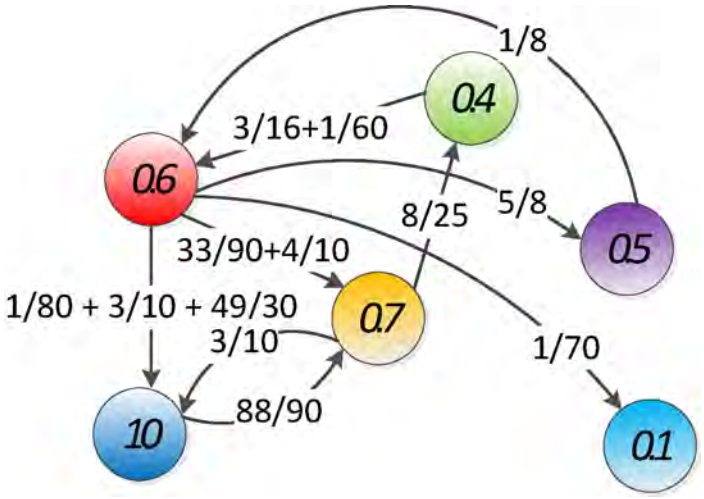


Figure 4-5. Lossy coupled network using easiness parameters. The number of edges is much less than the lossless coupled network.

u^1 has 8 in-neighbors, each neighbor v^1 influences u^1 with $w^1(v^1, u^1) = 0.1$. In network 2, $\theta^2(u^2) = 0.7$ and u^2 has 8 in-neighbors, each neighbor v^2 influences u^2 with $w^2(v^2, u^2) = 0.1$. The number of active neighbors to activate u is 1 and 7 in network 1 and 2, respectively. Intuitively, we can say that u is easier to be influenced in the first network. We quantify the influence *easiness* $\epsilon^i(u)$ that u is influenced in network i as the ratio between the total influence from friends and the threshold to be influenced.

$$\epsilon^i(u) = \frac{\sum_{v^i \in N^-(u^i)} w^i(v^i, u^i)}{\theta^i(u^i)}$$

We can use the influence easiness of a user in networks as the parameters of the condition 4–1.

Based on above observations, we couple multiple networks into one using parameters $\{\alpha^i(u)\}$. The vertex set is the set of users $V = \{u_1, u_2, \dots, u_n\}$. The threshold of vertex u is set to:

$$\theta(u) = \sum_{i=1}^{k} \alpha^i(u)\theta^i(u^i)$$

The weight of the edge (v, u) is:

$$w(v, u) = \sum_{i=1}^{k} \alpha^i(u) w^i(v^i, u^i)$$

where $w^i(v^i, u^i) = 0$ if there is no edge from v^i to u^i in i^{th} network.

Then the set of edges is $E = \{(v, u) | w(v, u) > 0\}$. Fig. 4-5 illustrates the loosy coupled network of the system of network in Fig. 4-3.

Besides the easiness, other metrics can be used with the same purpose. We enumerate here some other metrics.

Involvement. Nodes can be part of multiple social networks, but typically they are more involved in a few of them compared to others. We estimate *involvement* of a node v in a network G_i by measuring how strongly the 1-hop neighborhood v is connected and to what extent influence can propagate from one node to another in the 1-hop neighborhood. Formally we can define *involvement* of a node v in network G_i as:-

$$\sigma_v^i = \sum_{x,y \in \{N^i(v) \cup v\}} \frac{w^i(x, y)}{\theta_y^i}$$

where $N^i(v)$ is the set of all neighbors of v (both in-coming and out-going), $w^i(x, y)$ is the wt of edge (x, y) and θ_y^i is the threshold of y in G_i.

Average. This a baseline scheme just used for comparison purposes. We just take an average of the thresholds and edge-wts over all the networks, in which v belongs. So *average* of a node v in network i can be defined as

$$\alpha_v^i = \frac{1}{|P(v)|}$$

Next we show the relationship between the solution for the influence maximization problem in the lossy coupled network and the original system of networks. As discussed in the above observations, if the propagation process starts from the same set of users in the network system $G^{1...k}$ and the coupled network, then the active state of a user in G implies its active state in $G^{1...k}$. It means that if the set of users S activates β fraction

of users in G, it also activates at least β fraction of users in $G^{1...k}$. We have the following result.

Theorem 4.4. *When the lossy coupling scheme is used, if the set of users S activates β fraction of users in G, then it activates at least β fraction of users in $G^{1...k}$.*

4.5 Influence Relay

We propose the *influence relay* metric to quantify the role of users in propagating information. When the information is diffused in multiple networks, the information may flow within a single network or go through two or more networks. This brings out a series of concerns: how much information flows inside a network? how much information flows from one network to another? how much is the contribution of each network in the influence propagation? Once we can quantify these values, we can get insights into the influence diffusion process in multiple networks. Next, we define the *influence relay* metric and related concepts to measure these values.

Since we can use a single network to simulate the diffusion process in multiple networks, we first can measure the information flowing through each node in a single network. Suppose that the information breaks out from the seed set S in the network G, and stops after d hops with the set of influenced vertices $A^d(G, S)$. Intuitively, the influence relay of each vertex is the amount of influence it relays to other nodes after adopting the information. The more number of vertices it helps to influence, the higher its influence relay is. In addition, if it has strong influence on a node with high relay influence, it should also have high value of relay influence even it does not directly influence many vertices. For these reasons, we formally propose the influence relay metric $IR(\cdot)$ which is computed iteratively as below.

All inactive vertices have the influence relay of 0.

Each active vertex v without activated outgoing neighbors has the influence relay of 1. v does not activate any vertices, but it contributes itself as one active node to $A^d(G, S)$.

The influence relay of any other vertex u is computed based on the influence relay of its outgoing active neighbors. Specifically, the influence relay of u is:

$$IR(u) = 1 + \sum_{v \in N_u^+ \wedge h(u) < h(v)} \frac{w(u,v) IR(v)}{\sum_{z \in N_v^- \wedge h(z) < h(v)} w(z,v)} \tag{4–2}$$

where $h(u)$ is the hop at which u is activated.

In this formula, the influence relay of u is the total influence relay of vertices which are under u's influence. However, each vertex v of these ones is under the influence of many vertices. Among them, u contributes the impact of $w(u,v)$ to influence v, hence u is responsible for only $\frac{w(u,v)}{\sum_{z \in N_v^- \wedge h(z) < h(v)} w(z,v)}$ of v's influence relay. Moreover, we add 1 to the influence relay of u since u also contributes itself to the set of activated vertices.

We next present an efficient algorithm to compute the influence relay of all vertices. Since the influence relay of each node depends on its out-going neighbors which are activated later than it, we need to compute the influence relay of nodes in the reverse order of the diffusion process. We can construct the influence graph $IG_S = (V_S, E_S)$ from the seed set S to represent the diffusion process and compute the influence relay of all nodes in V_S. The vertex set V_S of n_S nodes is $A^d(G,S)$. There is an edge from u to v in E_S with the same weight $w(u,v)$ in G if u has passed the information to v, i.e., $u, v \in A^d(G,S)$ and $h(v) > h(u)$. The graph IG_S is a directed acyclic graph, thus we can compute the influence relay of all vertices in the reverse topological ordering of IG_S as described in *CIR* algorithm (Algorithm 11).

Proposition 4.4. *The influence graph IG_S caused by the seed set S in the network G is a directed acyclic graph.*

Proof. If IG_S has a cycle $u_1, u_2, \ldots, u_t, u_1$, then $h(u_1) < h(u_2) < \ldots < h(u_t) < h(u_1)$ (contradicted). □

Algorithm 11 Computing Influence Relay (CIR)

Require: A network G, a seed set S and the number of hops d.
Ensure: The influence relay IR of all vertices.

```
IG_S ← The influence graph caused by S on G
for each u ∈ V_S do
    IR(u) ← 0
end for
Compute the topological ordering u_1, u_2, ... , u_{n_S} of vertices in V_S
for i = n_S down to 1 do
    IR(u_i) ← IR(u_i) + 1
    total ← 0
    for each v ∈ N^-(u_i) do
        total ← total + w(v, u_i)
    end for
    for each v ∈ N^-(u_i) do
        IR(v) ← IR(v) + w(v,u_i) IR(u_i) / total
    end for
end for
Return IR
```

Lemma 6. *The* CIR *algorithm produces the influence relay for each activated vertex.*

Proof. We use induction to prove that the influence u_k is computed correctly after the loop $i = k$. Firstly, u_n is computed first and $IR(u_n) = 1$. This is correct since u_n is at the end of the topological ordering and does not have any activated outgoing neighbors.

Now, suppose that the influence relay of $u_n, u_{n-1}, \dots, u_{k+1}$ is computed correctly after the loop $i = k + 1$, we will prove that $IR(u_k)$ holds the influence relay of u_k after the loop $i = k$. Let $\{u_{i_1}, u_{i_2}, \dots, u_{i_p}\}$ be the set of activated outgoing neighbors of u_k which is activated later than u_k. Due to the construction of IG_S, $(u_k, u_{i_1}), (u_k, u_{i_2}), \dots, (u_k, u_{i_p})$ are edges in IG_S, hence $i_q > k$, $1 \leq q \leq p$, in the topological ordering. After the $i = i_q$ loop, $IR(u_k)$ will receive a value of $\frac{w(u_k,u_{i_q})IR(u_{i_q})}{\sum_{z \in N^-_{u_{i_q}} \wedge h(z) < h(u_{i_q})} w(z,u_{i_q})}$ from the influence relay of u_{i_q}. At loop $i = k$, $IR(u_k)$ is increased by 1 for u_k itself and equals the influence relay of u_k according to the Eq. 4–2. □

Time complexity. The topological ordering of a directed acyclic graph can be computed in linear time and the number of updates in the main loop equals to the number of edges of IG_S, so the *CIR* algorithm runs in linear time.

A crucial property of the new metric is that the total influence relay of seed vertices reflects the influence of the seed set as stated in Theorem 4.5.

Theorem 4.5. *The total influence relay of seeding vertices equals the total number of activated vertices.*

$$\sum_{u \in S} IR(u) = |A^d(G, S)|$$

Proof. The proof is based on an invariant of variables $IR(u_1), \dots, IR(u_n)$ in *CIR* algorithm. The information is propagated from the seed set, thus all seed vertices do not have incoming neighbors in IG_S and occupy smallest indices in the topological ordering. Let u_p be the highest index seed vertex. We will prove that after the loop $i = k + 1$ we have:

$$\sum_{j=1}^{k} IR(u_j) = n_S - k, \forall p \leq k \leq n_S$$

Before the loop $i = n$, it is obviously true. After the loop $i = k$, the value of variable $IR(u_{k+1})$ is increased by 1 and redistributed to its incoming neighbors, thus $\sum_{j=1}^{k-1} IR(u_j)$ equals $\sum_{j=1}^{k} IR(u_j)$ plus 1. It implies that $\sum_{j=1}^{k-1} IR(u_j) = n_S - (k-1)$ after the loop $i = k$.

After the loop $i = p + 1$, we have $\sum_{i=1}^{p} IR(u_i) = n_S - p$. At each loop $i = p$ down to $i = 1$, the value of $IR(u_i)$ is increased by 1. Thus, when the algorithm stops we have:

$$\sum_{u \in S} IR(u) = \sum_{i=1}^{p} IR(u_i) = n_S - p + p = |A^d(G, S)|$$

□

Theorem 4.5 implies that each vertex $u \in S$ contributes $IR(u)$ in propagating the influence over the network G. Now, suppose that G is the coupled network of multiple ones, we can sum up the influence relay of all seed vertices of a component network to obtain the contribution of that network. Furthermore, the total influence relay

of overlapping vertices indicates the amount information propagated back and forth between networks.

We can also adapt the influence relay metric to measure the support between networks in propagating information. Let's consider the case the information emerges from the seed vertices of one network, propagates to another networks via overlapping users, then comes back to the first network. With the support of other networks, the information is propagated further in the first network. If we consider the diffusion process in the coupled network and increase the influence relay by 1 only on activated vertices in the first network, we can quantify the support from other networks by the total influence relay which goes through other networks.

4.6 Experimental Evaluation

In this section, we show the experimental results for coupling schemes and use the coupling schemes to analyze the influence diffusion in multiple networks. Firstly, we compare lossless and lossy coupling schemes to measure the trade-off between the running time and the quality of solutions. Since the massive influence problem is NP-hard [17] in a single network, we use the greedy algorithm, which provides high quality solution, to find the solution after coupling networks. We also investigate the relationship between networks in the information diffusion to answer the following questions: (1) What is the role of overlapping users in the diffusion of the information? (2) How does a network get benefit from other networks to diffuse the information? (3) Can the diffusion on one network provide a burst of information in other networks? (4) What will we miss if we consider each network separately?

We ran all our experiments on Intel(R) Xeon(R) CPU W350 machine with a 12 GB RAM and a 2.93 GHz Quad-core processor. In all experiments, by default, the number of hops is $d = 4$ and the influence fraction $\beta = 0.8$.

Networks	#Nodes	#Edges	Avg. Degree
Twitter	48277	16304712	289.7
FSQ	44992	1664402	35.99
CM	40420	175692	8.69
Het	8360	15751	1.88
NetS	1588	2742	1.73

Table 4-1. Foursquare-Twitter and co-author network data-sets

4.6.1 Datasets

Real networks. We do experiments on two systems of networks: *Twitter* and *Foursquare* (*FSQ*) networks [49], and co-author networks in the area of Condensed Matter(CM) [43], High-Energy Theory(Het) [43], and Network Science (NetS) [42]. The statistics of networks are described in Table 4-1. While the overlapping users of the first dataset is provided in [49], we match overlapping users of the second one based on authors' names. The numbers of overlapping nodes of network pairs FSQ-Twitter, CM-Het, CM-NetS, and Het-NetS are 4100, 2860, 517, and 90, respectively. Moreover, while co-author networks have edge weights, FSQ-Twitter dataset only contains the network topologies. If the network does not have edge weights, we assign the weight of each edge randomly from 0 to 1. We then normalize the edge weights such that the total weight of in-coming edges is 1 for each node. This is suitable since the influence of user u on user v tends to be small if v is under the influence many friends. Finally, the threshold of each node is a random value from 0 to 1.

Synthesized networks. We also use synthesized networks generated by Erdos-Renyi random network model [26] to test networks with controlled parameters. There are two networks with 5000 nodes are formed by randomly connecting each pair of nodes with probability $p_1 = 0.0008$ and $p_2 = 0.006$. The average degrees, 8 and 60, reflect the diversity of network densities in the reality. Then, we select randomly f fraction of nodes

in the first network as overlapping nodes. The edge weights and node thresholds are assigned as above.

4.6.2 Comparison of Coupling Schemes

We first evaluate the effect of the coupling schemes on the running time and the quality of the found solutions when we use the greedy algorithm to solve MIP. As illustrated in Fig. 4-6, the algorithm provides larger seed sets but runs faster in lossy coupled networks than lossless coupled networks. In both Twitter-FSQ and co-author datasets, the seed sizes are smallest when the lossless coupling scheme is used. It is as expected since the lossless coupling scheme reserves all the influence information which is exploited later to solve MIP. However, the seed sizes are only a bit larger using the lossy coupling schemes. In the lossy coupling schemes, the information is only lost at overlapping users which occupies a small fraction the total number of users (roughly 5% in Twitter-FSQ and 7% in co-author networks). Thus, the effect of lossy coupling schemes on the solution quality is small especially when the seed sets are big to influence a large fraction of users. On the other hand, the algorithm runs much faster in lossy coupled networks with the factor up to 2 times in Twitter-FSQ and 4 times in co-author networks. The major disadvantages of the lossless coupling scheme is the doubled number of hops, the number of extra nodes and edges. In co-author dataset, the number of extra edges are relative high comparing to the total number of edges in all networks, so the speeding up factor is higher in co-author networks. We therefore can infer that the lossy coupling schemes work well on real datasets in which networks are sparse and the number of overlapping users is small. Next, we examine the effect of the number of overlapping users on the performance of the coupling schemes with the synthesized datasets. Fig. 4-7 demonstrates the results on two networks of size 5000 and different fraction of overlapping users f. The overlapping fraction significantly differentiate the coupling schemes in terms of both the solution quality and running time. When f is small, the seed sizes are quite close with all coupling schemes. But

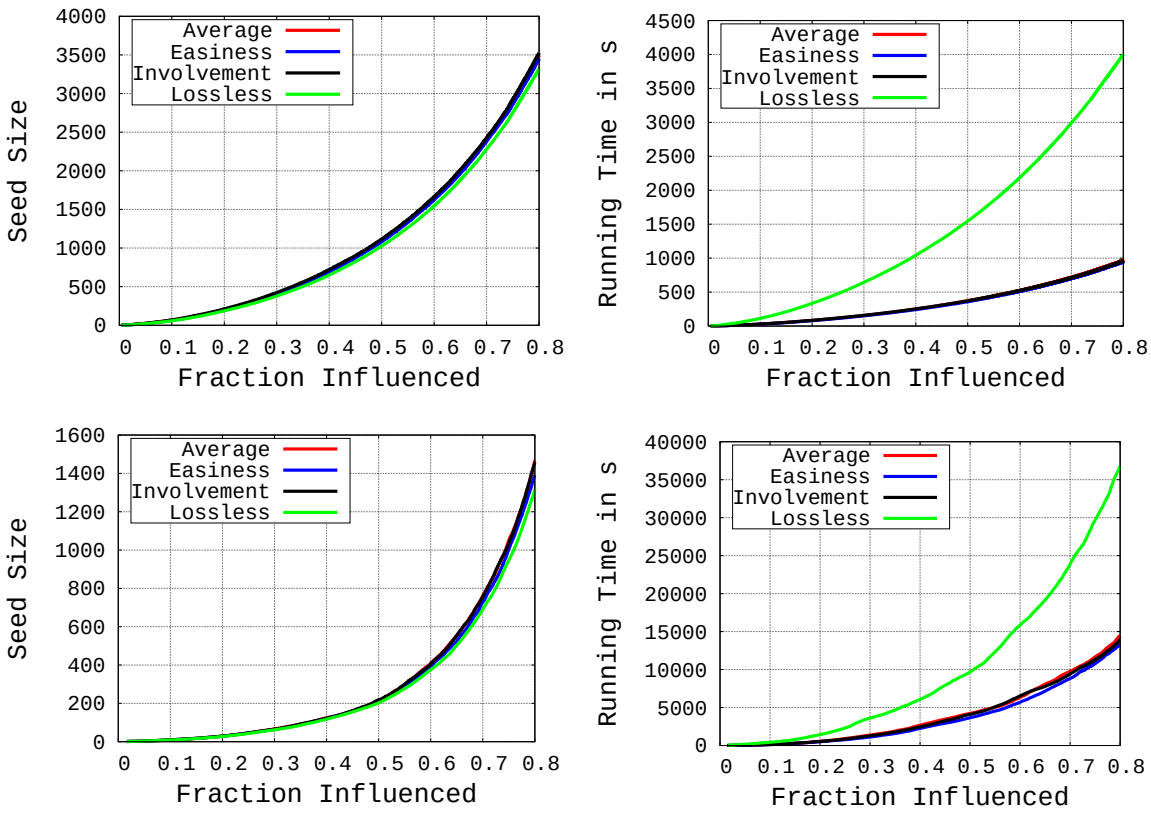


Figure 4-6. Comparing Coupling Schemes for Finding Minimum Seed Set on co-author Networks (upper figures) and on FSQ and Twitter (lower figures)

when f increases, the gap between schemes is bigger and bigger. The variation of f also reveals the effectiveness of the easiness lossy coupling scheme (the best) and the disgrace of the trivial average scheme (the worst) among the lossy ones. It is more interesting when we look at the running time. The running time in the lossless coupled networks is initially higher than in the lossy coupled networks but it gradually catches up and overtakes the later networks at $f = 0.4$. The key point is the size of the seed set. The larger f is, the larger the ratio between the seed size in lossless and lossy coupled networks is. As the running time depends on the seed size, the running time in the lossless coupled network reduces faster. Thus, we recommend to use lossless scheme when the overlapping fraction is large and the seed size is predicted to be small.

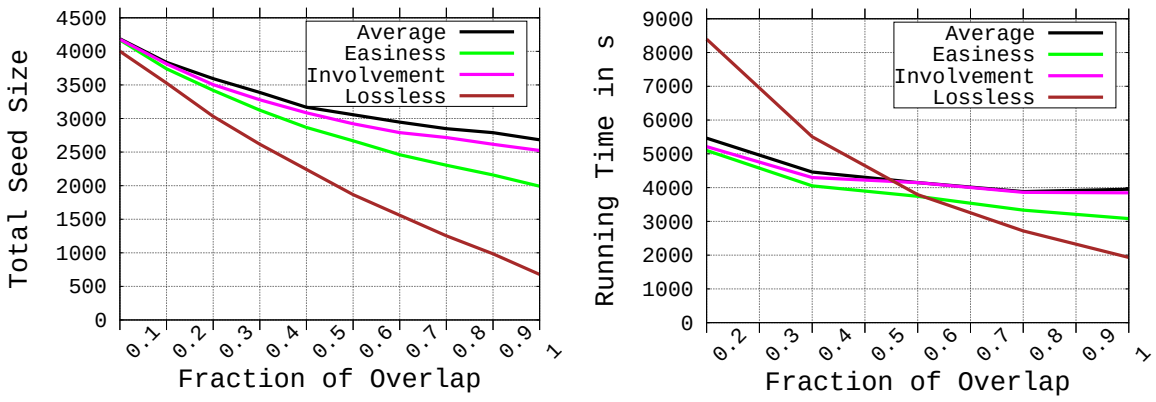
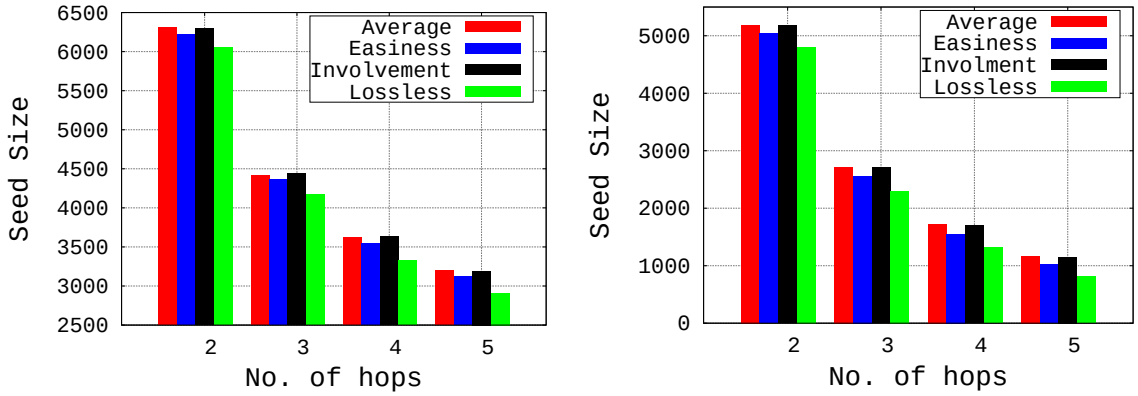


Figure 4-7. Comparing coupling schemes with different overlapping fraction f



A Co-author networks B FSQ and Twitter

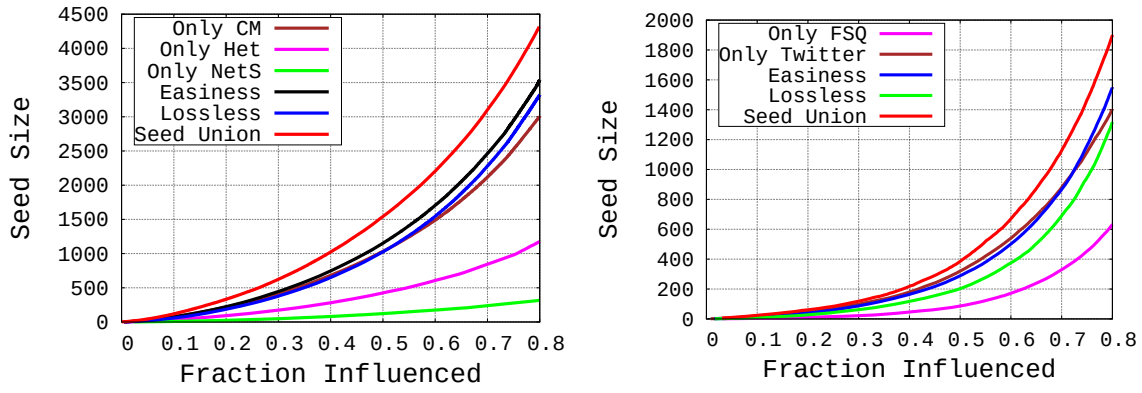
Figure 4-8. Comparing coupling schemes with different number of propagation hops d

Since the seed size is sensitive to the number of hops [24], we would like to evaluate coupling schemes with different propagation hops. Similar to single networks, Fig. 4-8 shows that the seed size decreases when we have larger number of propagation hops. However, the lossy coupling schemes deviate more and more from the lossless one in terms of the relative seed size when the number of hops increases. Let's consider the ratio of the seed sizes between the best lossy coupling scheme (the easiness one) and the lossless coupling scheme. It is 1.05 (1.1) and 1.5 (1.3) in co-author networks (FSQ-Twitter) with $d = 2$ and $d = 5$. The reason is that the lossy coupling schemes inherently bear the error which is accumulated and propagated after each hop.

4.6.3 Benefits of Coupled Network

Coupling schemes provide the mechanism to study multiple networks under a consistent view which helps to answer different concerns about the influence diffusion. For example, Fig. 4-6 shows a property that is similar to one in a single network: the seed size increases super linearly regarding to the influence fraction. It means that the gain per seed users is decreased when the circle of influence is broadened. Moreover, without the coupled network, we may need to find the seed set on each network to influence β fraction of all users and union them to obtain the seed set for the whole system. Fig. 4-9 clearly demonstrates that if we influence each network separately we would need a much larger seed set compared to what we need in the coupled network, no matter which type of coupling we use. The seed set found on the lossless coupled network almost has the same size with the largest seed set found in component networks in co-author dataset and even smaller in Twitter-FSQ. In co-author datasets, the size of the union set to influence 0.8 fraction of users is 24% and 30% larger the size of the seed sets found in lossless and lossy networks. These numbers are 23% and 47% in Twitter-FSQ. The reason is that the lossless (lossy) coupled network can capture (partially capture) the collaboration of networks to propagate the information and exploit it to reduce the seed size. When we find the seed set in each network separately, we ignore this property. As a consequence, we endure a penalty on the size of the union set which is high if networks can propagate the information well like Twitter and FSQ. Although we can use other methods to solve MIP without using coupling schemes, they may be more complicated and cause the seed size increase.

The coupling schemes not only help to solve MIP, it is also help to investigate other aspects of the influence diffusion in the system of networks. origin Due to the overlapping with other networks, we may underestimate the ability to diffuse the information of a specific network. It motivates us gauge the viral marketing potential of a network allowing the information to be propagate to back and forth to other networks.

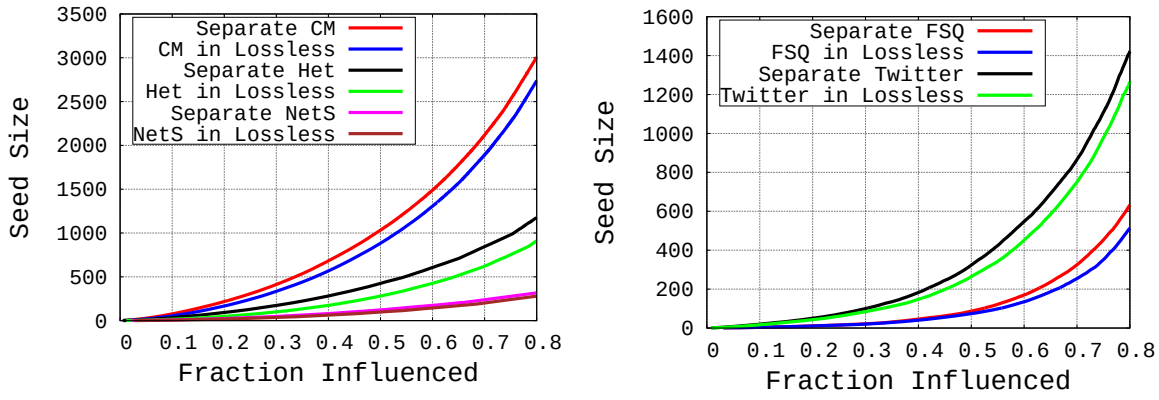


A Co-author networks

B FSQ and Twitter

Figure 4-9. The quality of seed sets with and without using the coupled network

Specifically, we use the greedy algorithm to find the smallest seed set to influence β fraction of the studied network' users in the lossless coupled network. Then we compare it to the seed set found in the traditional perspective – considering this network as a standalone one. As shown in Fig. 4-10, the seed size decreases up to 9%, 25%, 17%, and 26% in CM, Het, FSQ, and Twitter, respectively, when we consider these networks in the connection with other networks. The improvement in NetS is small due to the small number of overlapping users with other networks. It is also observed that the improvement ratio is higher for network with low conductance of influence in the case of FSQ and Twitter, two networks with the same number of users. When the network sizes are unbalanced, Het – the network with the smaller number of users seems to get better improvement ratio than the bigger network CM. The back and forth propagation of the information between networks is the base for the outside support of the target network. When the information is propagated from seed nodes in the target network, some nodes are activated in other networks due to the overlapping nodes. The information then is propagated further and even comes back to the target network, hence the number of influenced node in the target network is increased. Fig. 4-11 shows the amount of influence relay that other networks support the target network with $d = 4$ and $d = 8$ hops. The support is considerable and higher with the larger number of hops. When



A Co-author networks

B FSQ and Twitter

Figure 4-10. The quality of seed sets with and without using the coupled network

$d = 8$, the information has more chance to come back to the target network; the support is up to 2.2%, 5%, 8.3%, and 7.3% on the network CM, Het, Twitter, and FSQ. The support is higher if the information is easier to be propagated in component networks.

4.6.4 Bias in Selecting Seed Nodes

Here, we analyze the the seed set on the lossless coupled network to observe how much each network contributes towards the composition of the seed set and the set of influenced nodes. We mainly address two questions: (1) which network supports the propagation better and (2) whether there is a bias toward a network selecting seed nodes. Fig. 4-12 shows the fraction of selected nodes as well as influenced nodes in each network and the overlapping part. We can observe that overlapping nodes tend to be selected in both datasets. When the influenced fraction is 0.4, the fraction of overlapping seed nodes is around 24.9% and 25% on co-author and FSQ-Twitter networks, respectively. Note that only 5% (7%) total users of FSQ-Twitter (co-author networks) are overlapping users. This shows that overlapping users not only play the role as bridges for information to propagate between networks but also have high influence. As illustrated in Fig. 4-13, the contribution of the overlapping nodes in influencing other nodes is high, especially when β is small. Additionally, there is an unbalance between the number of selected seeds and influenced nodes in each

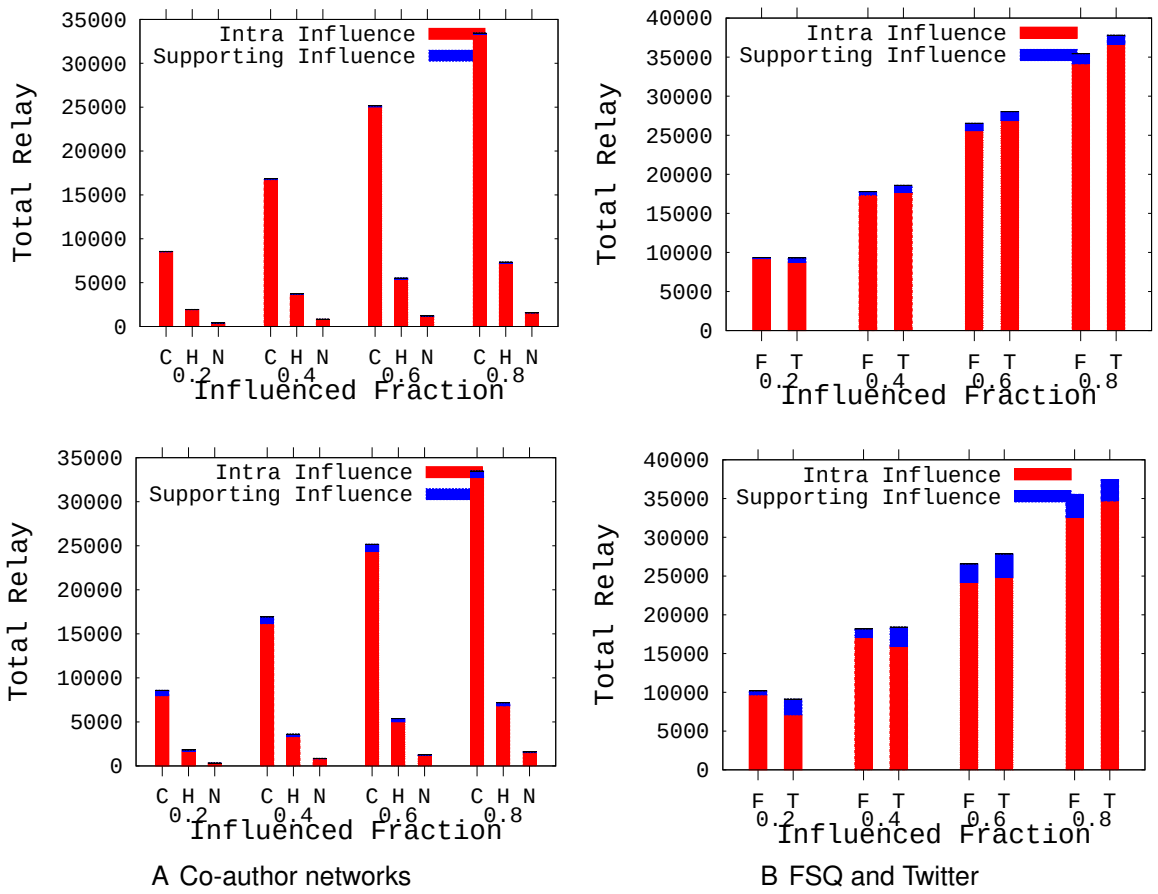


Figure 4-11. The support between networks on the influence propagation of a network with $d = 4$ (upper figures) and $d = 8$ (lower figures) hops. C, H, N, F, and T are the abbreviations of CM, Het, NetS, FSQ, and Twitter.

networks. In co-author dataset, the biggest network, CM, contributes a large number of seed nodes and influenced nodes. When $\beta = 0.8$, 76.7% of seed nodes and 80.5% of influenced nodes are from CM. In contrast, the number of seed nodes from FSQ is small but the number of influenced nodes in FSQ much higher than Twitter. Let consider the influence fraction of 0.4, 27% (without overlapping nodes) of seed nodes belong to FSQ while 70% of influenced nodes are in FSQ. After nodes in FSQ are almost influenced, the algorithm starts to select more nodes in Twitter to increase the influence fraction. It implies an important characteristic of multiple networks. If the information is easier to flow in one network, that network will attract and propagate more information inside. In

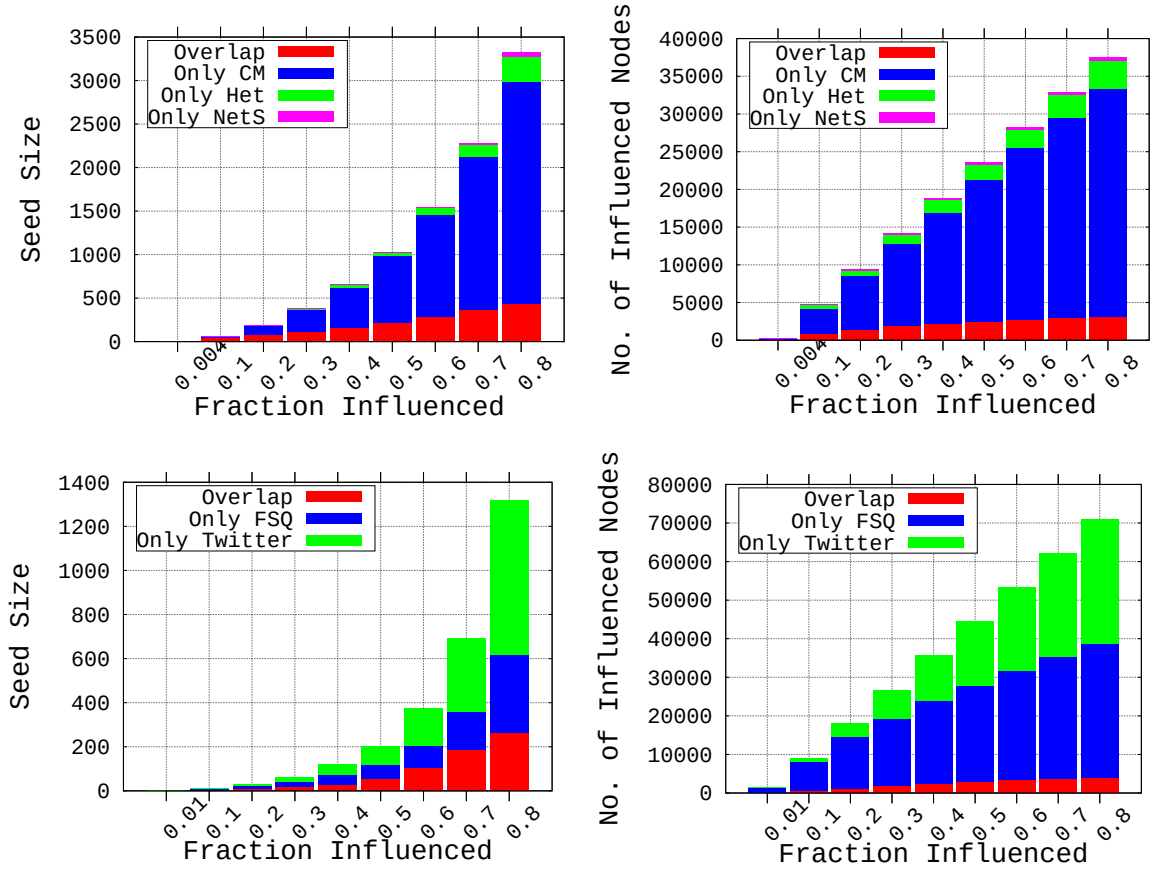


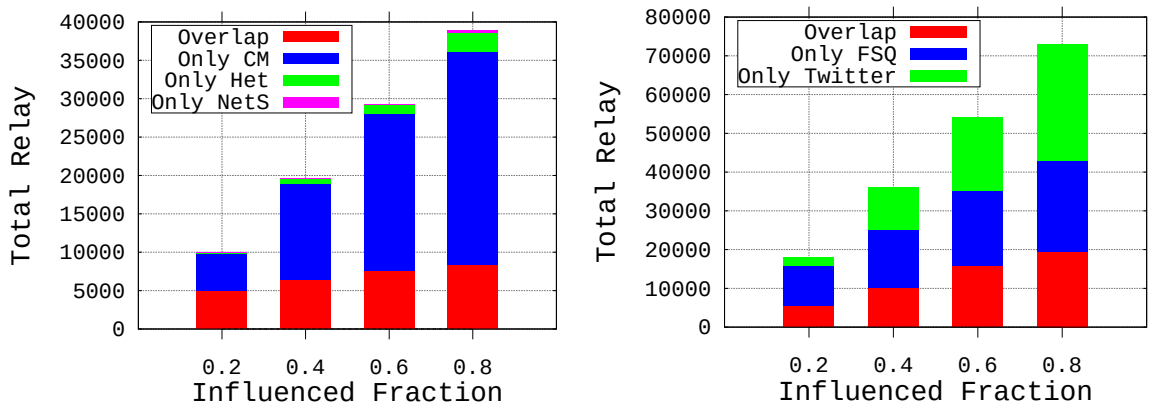
Figure 4-12. The bias in selecting seed nodes on synthesized networks (upper figures) and on FSQ and Twitter (lower figures)

the big picture, it provides hints for viral marking: overlapping nodes have high potential to target and some networks are more efficient to advertise than others.

4.7 Extensions to Other Cascading Models

In this section, we show that we can design lossless coupling schemes for some other well-known cascading models in each component network. As a consequence, top influential nodes can be identify under these models. In particular, we investigate two most popular stochastic diffusion models which are Stochastic Threshold model and Independent Cascading model [34].

- *Stochastic Threshold model.* This model is similar to the Linear Threshold model but the threshold $\theta^i(u^i)$ of each node u^i of G^i is a random value in the range $[0, \Theta^i(u^i)]$. Node u^i will be influenced when $\sum_{v^i \in N^-_{u^i}, v \in A} w^i(v^i, u^i) \geq \theta^i(u^i)$



A Co-author networks B FSQ and Twitter

Figure 4-13. The influence contribution of seed nodes from component networks

- *Independent Cascading model.* In this model, there are only edge weights representing the influence between users. Once node u^i of G^i is influenced, it has a single chance to influence its neighbor $v^i \in N^+(u^i)$ with probability $w^i(u^i, v^i)$.

For both models, we use the same approach of using user vertices, account vertices and the synchronization between user vertices and their account vertices. Specifically, the weight of edge (u^i, u^j), $0 \leq i \neq j \leq k$ will be $\Theta(u^j)$ for Stochastic Threshold model and 1 for Independent Cascading model. With this assignment, if u^i is influenced, u^j will be influenced with probability 1 in the next time step. The proof for the equivalence of the coupling scheme is similar to ones for Section 4.3.

4.8 Summary

In this chapter, we study the massive influence problem in multiple networks. To tackle the problem, we introduced novel coupling schemes to reduce the problem to a version on a single network. Then we design a new metric to quantify the flow of influence inside and between networks based on the coupled network. Exhaustive experiments provide new insights to the information diffusion in multiple networks.

In the future, we plan to investigate the problem on multiple networks with heterogeneous diffusion models. In particular, each network may have its own diffusion model, the question is how to represent them efficiently. Does there exist a method to couple them into one network?

CHAPTER 5
CONCLUSIONS

In this thesis, we study the problem of identifying granular nodes of the cascading propagation in networks. Under the cascading effect, these nodes have a strong impact over the network. It is crucial to detect such nodes to serve various purposes e.g. the economical gain. When the studied networks are social networks, we can use nodes as the target for advertising. On the other hand, if networks are infrastructure on like power network, communication networks, etc., we can protect these nodes from being attacked. For each kind of networks, we propose efficient strategies to find such group of nodes.

In interdependent infrastructure network, we introduce a new centrality for coupled networks and utilize it to detect most vulnerable nodes. In addition, a efficient greedy framework is proposed where the pure greedy and centrality measure are combined to provide a better solution in shorter time. In multiple online social networks, we design a novel framework to find top influential users. In particular, novel coupling schemes are designed to reduce the problem on multiple networks to one on a network. As a consequence, we can apply existing solutions for a network to find most influential nodes in multiple networks. It is a crucial connection which shows that solving the problem on coupled networks is as easy as on the single network. We believe that the coupling schemes can be extended to other models. Finally, we investigate the cascading failure under load redistribution model in power networks. A new cascading centrality is designed specifically load redistribution model and can be used to detect most critical nodes efficiently. Moreover, we propose the cooperating attack strategy to evaluate the weakness of networks even when it is designed to tolerate node failures.

REFERENCES

[1] "Michael jackson on tmz, iran on twitter." http://www.blogher.com/spreading-news, 2009.

[2] "Overlap Among Major Social Network Services." http://www.tomhcanderson.com/2009/07/09/overlap-among-major-social-network-services/, 2009.

[3] "216 Social Media and Internet Statistics." http://thesocialskinny.com/216-social-media-and-internet-statistics-september-2012/, 2012.

[4] "99 New Social Media Stats for 2012." http://thesocialskinny.com/99-new-social-media-stats-for-2012/, 2012.

[5] Albert, R., Albert, I., and Nakarado, G. L. "Structural vulnerability of the North American power grid." *Phys. Rev. E* 69 (2004).2.

[6] Albert, R., Jeong, H., and Barabasi, A.L. "Error and attack tolerance of complex networks." *Nature* 406 (2000).6794: 378–382.

[7] Albert, Réka, Albert, István, and Nakarado, Gary L. "Structural vulnerability of the North American power grid." *Physical Review E* 69 (2004).2: 025103.

[8] Arulselvan, Ashwin, Commander, Clayton W., Elefteriadou, Lily, and Pardalos, Panos M. "Detecting critical nodes in sparse graphs." *Comput. Oper. Res.* 36 (2009): 2193–2200.

[9] Barabasi, Albert-Laszlo and Albert, Reka. "Emergence of Scaling in Random Networks." *Science* 286 (1999).5439: 509–512.

[10] Borgatti, Stephen P. "Identifying sets of key players in a social network." *Comput. Math. Organ. Theory* 12 (2006).1: 21–34.

[11] Borgatti, Stephen P. and Everett, Martin G. "A Graph-theoretic perspective on centrality." *Social Networks* 28 (2006).4: 466 – 484.

[12] Bornholdt, Stefan and Schuster, Heinz Georg, eds. *Handbook of Graphs and Networks: From the Genome to the Internet*. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[13] Brandes, Ulrik and Erlebach, Thomas. *Network Analysis: Methodological Foundations*. Springer, 2005.

[14] Brown, Jacqueline Johnson and Reingen, Peter H. "Social Ties and Word-of-Mouth Referral Behavior." *Journal of Consumer Research* 14 (1987).3: pp. 350–362.

[15] Buldyrev, Sergey V, Parshani, Roni, Paul, Gerald, Stanley, H Eugene, and Havlin, Shlomo. "Catastrophic cascade of failures in interdependent networks." *Nature* 464 (2010).7291: 1025–1028.

[16] Buldyrev, Sergey V., Shere, Nathaniel W., and Cwilich, Gabriel A. “Interdependent networks with identical degrees of mutually dependent nodes.” *Phys. Rev. E* 83 (2011): 016112.

[17] Chen, Ning. “On the approximability of influence in social networks.” *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*. SODA ’08. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008, 1029–1037.

[18] Chen, Wei, Wang, Chi, and Wang, Yajun. “Scalable influence maximization for prevalent viral marketing in large-scale social networks.” *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’10. New York, NY, USA: ACM, 2010, 1029–1038.

[19] Chen, Wei, Yuan, Yifei, and Zhang, Li. “Scalable Influence Maximization in Social Networks under the Linear Threshold Model.” *Proceedings of the 2010 IEEE International Conference on Data Mining*. ICDM ’10. Washington, DC, USA: IEEE Computer Society, 2010, 88–97.

[20] Chlebk, Miroslav and Chlebkov, Janka. “Complexity of approximating bounded variants of optimization problems.” *Theoretical Computer Science* 354 (2006).3: 320 – 338.

[21] Church, Richard L., Scaparra, Maria P., and Middleton, Richard S. “Identifying Critical Infrastructure: The Median and Covering Facility Interdiction Problems.” *Annals of the Association of American Geographers* 94 (2004).3: pp. 491–502.

[22] Crucitti, Paolo, Latora, Vito, Marchiori, Massimo, and Rapisarda, Andrea. “Error and attack tolerance of complex networks.” *Physica A: Statistical Mechanics and its Applications* 340 (2004).1: 388–394.

[23] Csardi, Gabor and Nepusz, Tamas. “The igraph software package for complex network research.” *InterJournal* Complex Systems (2006): 1695.

[24] Dinh, Thang N., Nguyen, Dung T., and Thai, My T. “Cheap, easy, and massively effective viral marketing in social networks: truth or fiction?” *Proceedings of the 23rd ACM conference on Hypertext and social media*. HT ’12. New York, NY, USA: ACM, 2012, 165–174.

[25] Dinh, Thang N., Xuan, Ying, Thai, My T., Park, E. K., and Znati, Taieb. “On Approximation of New Optimization Methods for Assessing Network Vulnerability.” *INFOCOM*. IEEE, 2010, 2678–2686.

[26] Erdos, Paul and Rényi, Alfréd. “On the evolution of random graphs.” *Bull. Inst. Internat. Statist* 38 (1961).4: 343–347.

[27] Gao, Jianxi, Buldyrev, Sergey V, Stanley, H Eugene, and Havlin, Shlomo. “Networks formed from interdependent networks.” *Nature Physics* 8 (2011).1: 40–48.

[28] Goldenberg, J., Libai, B., and Muller, E. “Talk of the network: A complex systems look at the underlying process of word-of-mouth.” *Marketing letters* 12 (2001).3: 211–223.

[29] Grubesic, Tony H., Matisziw, Timothy C., Murray, Alan T., and Snediker, Diane. “Comparative Approaches for Assessing Network Vulnerability.” *International Regional Science Review* 31 (2008).1: 88–112.

[30] Hines, Paul, Blumsack, Seth, Cotilla Sanchez, E, and Barrows, Clayton. “The topological and electrical structure of power grids.” *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*. IEEE, 2010, 1–10.

[31] Hopcroft, John and Tarjan, Robert. “Algorithm 447: efficient algorithms for graph manipulation.” *Commun. ACM* 16 (1973).6: 372–378.

[32] Huang, Xuqing, Gao, Jianxi, Buldyrev, Sergey V., Havlin, Shlomo, and Stanley, H. Eugene. “Robustness of interdependent networks under targeted attack.” *Phys. Rev. E* 83 (2011): 065101.

[33] Hwang, W., Cho, and Ramanathan, M. “Bridging Centrality: Identifying Bridging Nodes in Scale-free Networks.” *Technical Report, Department of CSE, University at Buffalo*. 2006.

[34] Kempe, David, Kleinberg, Jon, and Tardos, Éva. “Maximizing the spread of influence through a social network.” *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’03. New York, NY, USA: ACM, 2003, 137–146.

[35] ———. “Influential nodes in a diffusion model for social networks.” *Proceedings of the 32nd international conference on Automata, Languages and Programming*. ICALP’05. Berlin, Heidelberg: Springer-Verlag, 2005, 1127–1138.

[36] Kinney, Reka, Crucitti, Paolo, Albert, Reka, and Latora, Vito. “Modeling cascading failures in the North American power grid.” *The European Physical Journal B-Condensed Matter and Complex Systems* 46 (2005).1: 101–107.

[37] Leskovec, Jure, Krause, Andreas, Guestrin, Carlos, Faloutsos, Christos, VanBriesen, Jeanne, and Glance, Natalie. “Cost-effective outbreak detection in networks.” *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’07. New York, NY, USA: ACM, 2007, 420–429.

[38] Liu, Xingjie, He, Qi, Tian, Yuanyuan, Lee, Wang-Chien, McPherson, John, and Han, Jiawei. “Event-based social networks: linking the online and offline social worlds.” *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD ’12. New York, NY, USA: ACM, 2012, 1032–1040.

[39] Luciano, Rodrigues, F.A., Travieso, G., and Boas, V. P. R. “Characterization of complex networks: A survey of measurements.” *Advances in Physics* 56 (2007).1: 167–242.

[40] Motter, Adilson E and Lai, Ying-Cheng. “Cascade-based attacks on complex networks.” *Physical Review E* 66 (2002).6: 065102.

[41] Murray, Alan T., Matisziw, Timothy C., and Grubesic, Tony H. “A Methodological Overview of Network Vulnerability Analysis.” *Growth and Change* 39 (2008).4: 573–592.

[42] Newman, M. E. J. “Finding community structure in networks using the eigenvectors of matrices.” *Phys. Rev. E* 74 (2006): 036104.

[43] Newman, Mark EJ. “The structure of scientific collaboration networks.” *PNAS* 98 (2001).2: 404–409.

[44] Parshani, R., Rozenblat, C., Ietri, D., Ducruet, C., and Havlin, S. “Inter-similarity between coupled networks.” *EPL (Europhysics Letters)* 92 (2010).6: 68002.

[45] Parshani, Roni, Buldyrev, Sergey V, and Havlin, Shlomo. “Interdependent networks: reducing the coupling strength leads to a change from a first to second order percolation transition.” *Physical Review Letters* 105 (2010).4: 048701.

[46] Rosas-Casals, Marti, Valverde, Sergi, and Solé, Ricard V. “Topological vulnerability of the European power grid under errors and attacks.” *International Journal of Bifurcation and Chaos* 17 (2007).07: 2465–2475.

[47] Rosato, V, Issacharoff, L, Tiriticco, F, Meloni, S, Porcellinis, S De, and Setola, R. “Modelling interdependent infrastructures using interacting dynamical models.” *International Journal of Critical Infrastructures* 4 (2008).1/2: 63.

[48] Schneider, Christian M, Araujo, Nuno A M, Havlin, Shlomo, and Herrmann, Hans J. “Towards designing robust coupled networks.” *Minerva* (2011): 1–7.

[49] Shen, Yilin, Dinh, Thang N., Zhang, Huiyuan, and Thai, My T. “Interest-matching information propagation in multiple online social networks.” *Proceedings of the 21st ACM international conference on Information and knowledge management*. CIKM ’12. New York, NY, USA: ACM, 2012, 1824–1828.

[50] Strang, Gilbert. *Linear Algebra and Its Applications*. Brooks Cole, 1988, 3 ed.

[51] Vazirani, Vijay V. *Approximation algorithms*. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

[52] Wang, Feng, Camacho, Erika, and Xu, Kuai. “Positive influence dominating set in online social networks.” *Combinatorial Optimization and Applications*. Springer, 2009. 313–321.

[53] Wang, Jian-Wei and Rong, Li-Li. “Cascade-based attack vulnerability on the US power grid.” *Safety Science* 47 (2009).10: 1332 – 1336.

[54] Wang, Wenkai, Cai, Qiao, Sun, Yan, and He, Haibo. “Risk-aware attacks and catastrophic cascading failures in us power grid.” *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, 1–6.

[55] Watts, Duncan J and Strogatz, Steven H. “Collective dynamics of small-worldnetworks.” *nature* 393 (1998).6684: 440–442.

[56] Wu, Zhi-Xi, Peng, Gang, Wang, Wen-Xu, Chan, Sammy, and Wong, Eric Wing-Ming. “Cascading failure spreading on weighted heterogeneous networks.” *Journal of Statistical Mechanics: Theory and Experiment* 2008 (2008).05: P05013.

[57] Xia, Yongxiang, Tse, Chi K., Tam, Wai M., Lau, Francis C. M., and Small, Michael. “Scale-free user-network approach to telephone network traffic analysis.” *Phys. Rev. E* 72 (2005): 026116.

[58] Yagan, Osman, Qian, Dajun, Zhang, Junshan, and Cochran, Douglas. “Information diffusion in overlaying social-physical networks.” *Information Sciences and Systems (CISS), 2012 46th Annual Conference on*. 2012, 1 –6.

[59] Zhao, Liang, Park, Kwangho, and Lai, Ying-Cheng. “Attack vulnerability of scale-free networks due to cascading breakdown.” *Physical review E* 70 (2004).3: 035101.

[60] Zhao, Liang, Park, Kwangho, Lai, Ying-Cheng, and Ye, Nong. “Tolerance of scale-free networks against attack-induced cascades.” *Physical Review E* 72 (2005).2: 025104.

[61] Zhu, Xu, Yu, Jieun, Lee, Wonjun, Kim, Donghyun, Shan, Shan, and Du, Ding-Zhu. “New dominating sets in social networks.” *Journal of Global Optimization* 48 (2010).4: 633–642.

[62] Zou, Feng, Zhang, Zhao, and Wu, Weili. “Latency-Bounded Minimum Influential Node Selection in Social Networks.” *WASA*. 2009, 519–526.

BIOGRAPHICAL SKETCH

Dung T. Nguyen received the BS degree in Information Technology from Hanoi University of Science and Technology, Hanoi, Vietnam in 2008. He is currently a PhD student at the Department of Computer and Information Science and Engineering, University of Florida, under the supervision of Dr. My T. Thai. His areas of interest are viral marketing on online social networks, vulnerability and cascading failures on coupled networks, and approximation algorithms for network optimization problems.

DEPARTMENT OF DEFENSE

DEFENSE THREAT REDUCTION
AGENCY
8725 JOHN J. KINGMAN ROAD
STOP 6201
FORT BELVOIR, VA 22060
ATTN: A. LYALIKOV

DEFENSE TECHNICAL
INFORMATION CENTER
8725 JOHN J. KINGMAN ROAD,
SUITE 0944
FT. BELVOIR, VA 22060-6201
ATTN: DTIC/OCA

DEPARTMENT OF DEFENSE CONTRACTORS

QUANTERION SOLUTIONS, INC.
1680 TEXAS STREET, SE
KIRTLAND AFB, NM 87117-5669
ATTN: DTRIAC